UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Titulació:

Màster Universitari en Enginyeria Industrial

Alumne:

Jordi Allepuz Raja

Enunciat TFM:

Estudi de comparació d'heurístiques per resoldre el problema de flux de tasques amb temps de procés estocàstics

Director del TFM:

José María Sallán

Convocatòria de lliurament del TFM:

Convocatòria extraordinària Abril 2019

Sumari

Sumari de figures

Sumari de taules

1. Introducció

1.1. Objecte

El principal objectiu d'aquest estudi és abordar el problema de flux de tasques amb temps estocàstics a través de diferents heurístiques, per tal de comparar l'efectivitat de cadascuna d'elles a l'hora de resoldre'l, en termes de qualitat de la solució, és a dir, proximitat a una solució òptima del problema i, per altra banda, en termes d'economia de computació, o dit d'una altra manera, avaluar quines heurístiques permeten resoldre el problema d'una manera més ràpida i eficient.

1.2. Abast

El marc de treball de l'estudi queda limitat des de dos fronts. Per una banda, el problema de flux de tasques és un camp d'estudi molt ampli. En aquest cas, s'abordarà el problema amb temps estocàstics, és a dir, els temps amb els que es treballaran seguiran una distribució de probabilitat concreta, que en aquest estudi serà una distribució de probabilitat normal.

Per altra banda, el problema pot ser resolt via diferents mètodes. En aquest estudi es treballaran diferents heurístiques combinades: Simulated Annealing (SA), Tabu Search (TS), heurística NEH i heurística CDS. Cadascuna d'elles serà desenvolupada en el llenguatge de programació R per a resoldre el problema de flux de tasques.

1.3. Especificacions bàsiques

Els algoritmes que es faran servir durant l'estudi conformaran un algoritme ILS combinat amb la simulació dels valors de la funció objectiu. Per altra banda, les instàncies que s'estudiaran per a resoldre el problema són aquelles que són àmpliament usades a la bibliografia consultada. Aquestes instàncies seran extretes del benchmark de E.Taillard, amb valors de dimensió de 5, 10 i 20 màquines i, per altra banda, amb valors de 20,50 i 100 tasques.

1.4. Justificació

El problema del flux de tasques, més conegut com a problema del taller mecànic, és un problema clàssic d'optimització combinatòria, el qual ha estat estudiat àmpliament per a instàncies deterministes. Multitud d'algoritmes han estat desenvolupats per a resoldre'l: heurístiques de

simplificació, algoritmes de cerca local o els algoritmes genètics en són exemples. L'evolució natural d'aquest problema passa per l'estudi del comportament d'aquest al introduir variabilitat a través d'instàncies no deterministes. Aquest és un camp poc estudiat, per la qual cosa, aquest estudi suposa un repte.

2. Estat de l'art

Per tal d'assolir els objectius del present estudi, així com comprendre la complexitat del problema que s'aborda i la metodologia per resoldre'l, s'ha estudiat el seu rerefons teòric amb profunditat. En un primer estadi de l'estudi, les preguntes que poden sorgir són molt bàsiques i comunes a qualsevol problema matemàtic o tècnic: Quin tipus de problema s'afronta? Com es pot resoldre? Quina complexitat presenta? Quins recursos es necessiten per resoldre'l?. Durant el desenvolupament de l'estudi s'aniran plantejant les respostes a aquestes preguntes i també sorgiran de noves, algunes de les quals quedaran sense resoldre. Aquesta última afirmació no és trivial, ja que actualment el problema del flux de tasques, així com molts problemes relacionats, encara no han pogut ser resolts. El camp de la optimització combinatòria és una branca de la matemàtica que viu en constant creixement des de l'auge de la informàtica a la segona meitat del segle XX i, a dia d'avui, és un camp on s'hi està desenvolupant una gran activitat investigadora. Durant la primera part de l'estudi, es realitzarà una introducció teòrica als problemes d'optimització combinatòria i s'aprofundirà en el problema del flux de tasques que ens incumbeix. Un cop establert el marc teòric en el que es basarà l'estudi, es passarà a una segona etapa de l'estudi on s'aplicaran els conceptes exposats mitjançant la programació del codi en R corresponent, per tal de contrastar l'efectivitat de les diferents metodologies estudiades.

2.1. Optimització combinatòria

El camp de la optimització combinatòria és una branca de la matemàtica aplicada que abasta no només la matemàtica sinó també gran part de la investigació en ciències computacionals, informàtica i diferents camps de l'enginyeria. A grans trets, la optimització combinatòria agrupa problemes amb un conjunt finit de solucions, d'on es pretén trobar una solució òptima d'entre aquestes. Depenent del tipus de problema, aquesta solució pot ser exacta o no. Des d'un punt de vista teòric, tots els problemes d'optimització combinatòria tenen una solució exacta, però tal com es comprovarà posteriorment, trobar la solució exacta resultarà una tasca certament complicada i en la majoria de casos impossible, degut a la gran quantitat de solucions possibles que pot tenir el problema. Aleshores, quan parlem de solucionar un problema d'optimització combinatòria, no pretenem trobar la solució al problema, sinó una bona solució a aquest. És a dir, una solució propera a l'òptim.

Per definir de manera formal un problema d'optimització combinatòria, primer és necessari definir el concepte **instància** i **funció objectiu** [6]:

- **Instància:** Formada per un conjunt finit de solucions – F – i una funció objectiu – c – d'on volem determinar per quin punt $f \in F$ es satisfà:

$$c(f) \leq c(y) \quad per\ a\ tota\ y \in F$$

Al punt f se l'anomena òptim i tal com s'ha explicat anteriorment, de manera general, mai s'hi arribarà, però sí es pot trobar una solució propera.

- **Funció objectiu:** Valor numèric associat a una solució $f \in F$, el qual ens defineix la qualitat de la solució.

El conjunt d'instàncies formades de manera anàloga formen el problema d'optimització combinatòria. Cada una d'aquestes instàncies té associada una dimensió n, la qual ens determinarà la dificultat del problema, tal com veurem al punt 2.1.1.

2.1.1. Classificació dels problemes d'optimització combinatòria

La classificació d'aquest tipus de problemes ve determinada per la dimensió del problema n, la qual determinarà de manera clau la dificultat del problema. Els mètodes de resolució dels problemes combinatoris es basen en la iteració de possibles solucions, on es calcula la funció objectiu de cada solució i es compara amb altres solucions trobades anteriorment per determinar si la qualitat d'aquesta última és millor o no que les anteriors. Per tant, l'augment de la dimensió del problema fa augmentar ràpidament el nombre de possibles solucions d'aquest i, conseqüentment, serà més difícil trobar una solució propera a la òptima dins del conjunt de solucions possibles.

Un concepte important a tenir en compte a l'hora de classificar els problemes combinatoris és el de **temps de computació**, o més concretament el de **temps polinòmic**. En aquest moment, la optimització combinatòria fa el salt al món de la ciència computacional.

- **Temps de computació:** És el temps necessari per executar un algoritme. Tal com s'ha comentat anteriorment, aquest temps va directament lligat a la dimensió del problema.
- **Temps polinòmic:** Quan el temps de computació d'un algoritme es pot definir mitjançant una expressió polinòmica depenent de n. La notació del temps polinòmic és la següent [6]:

$$O(\log n) \rightarrow Temps\ logarítmic$$
$$O(n) \rightarrow Temps\ lineal$$
$$O(n^2) \rightarrow Temps\ quadràtic$$
$$O(n^3) \rightarrow Temps\ cúbic$$

Tots aquells problemes que no poden ser resolts per un algoritme en un temps polinòmic, són resolts per algoritmes en un temps exponencial. Sembla obvi doncs que, per a instàncies amb dimensions mitjanes i grans, aquests últims necessiten d'un temps de computació tan elevat que no es poden resoldre i només en podrem obtenir solucions aproximades.

Altres conceptes a tenir en compte són els de **algoritme determinista** i **algoritme no determinista**[7]:

- **Algoritme determinista:** És aquell algoritme que segueix un únic camí de computació. És a dir, per una mateixa entrada, sempre obtindrem la mateixa sortida.
- **Algoritme no determinista:** És aquell algoritme que no segueix un únic camí de computació, sinó que pot trobar-se en punts de decisió. En aquest cas, l'algoritme segueix un arbre de decisió i pot acceptar o no una decisió seguint un criteri d'acceptació.

Finalment, els últims conceptes a tenir en compte són la **verificació de la solució** i la **quantificació de la solució** [8]**:**

- **Verificació de la solució:** Contenen totes aquelles operacions necessàries per a verificar si un candidat a solució pertany o no al conjunt de possibles solucions.
- **Quantificació de la solució:** Contenen totes aquelles operacions necessàries per a obtenir un valor de la funció objectiu d'una solució determinada.

Ambdós conceptes van estretament lligats amb el temps de computació, ja que la dificultat per verificar i quantificar una solució, serà determinant en el temps d'execució d'un algoritme.

Un cop definits aquests conceptes bàsics, es pot passar a classificar el problemes d'optimització combinatòria. Segons el tipus d'algoritme necessari per a resoldre el problema i el temps de computació emprat, la complexitat dels problemes d'optimització combinatòria es pot classificar de la manera següent [9]:

- **Problemes P:** Són tots aquells problemes que poden ser resolts en un temps polinòmic per un algoritme determinista.
- **Problemes NP:** Són tots aquells problemes que poden ser verificats en un temps polinòmic per un algoritme determinista o solucionats per un algoritme no determinista en un temps polinòmic.
- **Problemes NP-Complet:** Són tots aquells problemes que no poden ser resolts en un temps polinòmic, però poden reduir-se o simplificar-se a un altre problema que pot ser resolt en temps polinòmic. En altres paraules, són els problemes més difícils dins del conjunt de problemes NP.
- **Problemes NP-Hard:** Són tot aquells problemes que no poden ser resolts en un temps polinòmic i no tenen la possibilitat de reduir-se a un problema NP-Complet.

Gràficament, es pot representar la classificació d'aquest tipus de problemes tal com es mostra a la Figura 1.

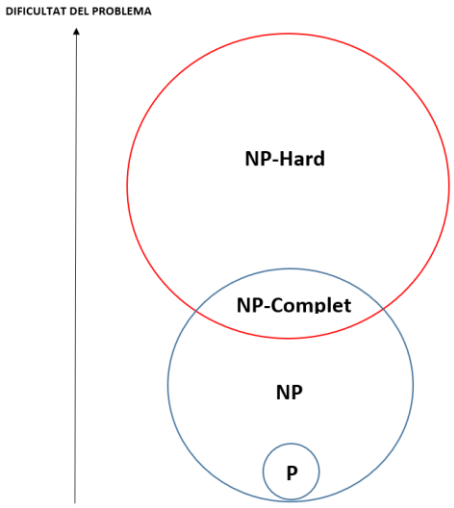

Figura 1. Classificació dels problemes d'optimització combinatòria segons la seva dificultat.

Actualment, encara està per determinar si tots els problemes de tipus NP pertanyen al tipus P. Aquest és un dels majors, sinó el major, dels problemes per resoldre actualment en el camp de la ciència computacional. La recerca actual intenta trobar algoritmes capaços de solucionar problemes NP-Complets o NP-Hard en un temps polinòmic. Fins el moment, encara no s'ha aconseguit trobar cap algoritme que sigui capaç de fer-ho, però, en el cas que es trobés, es podria concloure que tots els problemes NP serien NP-Complets, i conseqüentment tots els problemes NP serien del tipus P. Aquest seria un gran avenç en el camp de la optimització combinatòria, ja que voldria dir que s'hauria trobat la manera de resoldre problemes que fins ara eren impossibles de solucionar.

2.1.2. Problemes clàssics de l'optimització combinatòria

A continuació s'analitzaran alguns exemples típics de problemes d'optimització combinatòria. Concretament, es treballarà el "problema del viatger", conegut en anglès com a Travelling Salesman Problem (TSP), el "problema d'optimització de rutes", conegut com a Vehicle Routing Problem (VRP) i el "problema de la motxilla" o Knapsack Problem (KP). El problema que ens aborda en aquest estudi és considerat també un problema clàssic d'optimització combinatòria, però serà analitzat amb més profunditat a l'apartat 2.2.

2.1.2.1. Travelling Salesman Problem – TSP

El problema del viatger es pot definir formalment de la següent manera [10].

Donat un conjunt de nodes:

$$V = \{n_1, n_2, n_3 \dots n_i\} \quad (i \geq 3)$$

La unió dels nodes forma una xarxa o graf, amb un cost o pes entre nodes definit com:

$$\delta(n_{i_1}, n_{i_2}) \quad \rightarrow \quad Considerant\ \delta(n_{i_1}, n_{i_2}) = \delta(n_{i_2}, n_{i_1})$$

L'objectiu del TSP es trobar el mínim cost per a visitar tots els nodes. En altres paraules, de manera informal es pot definir el TSP com un viatger o comerciant que ha de visitar un nombre determinat de ciutats. Aquestes ciutats estan unides entre elles (no necessàriament tots els nodes estan units entre si) i per anar d'una a l'altra s'ha de recórrer una distància determinada prèviament (cost o pes). Com es pot observar, la definició del problema és força general i en ella es poden encabir multitud de problemes dins l'àmbit industrial o de l'enginyeria en general. A la Figura 2[12] es pot observar un exemple gràfic del problema TSP.

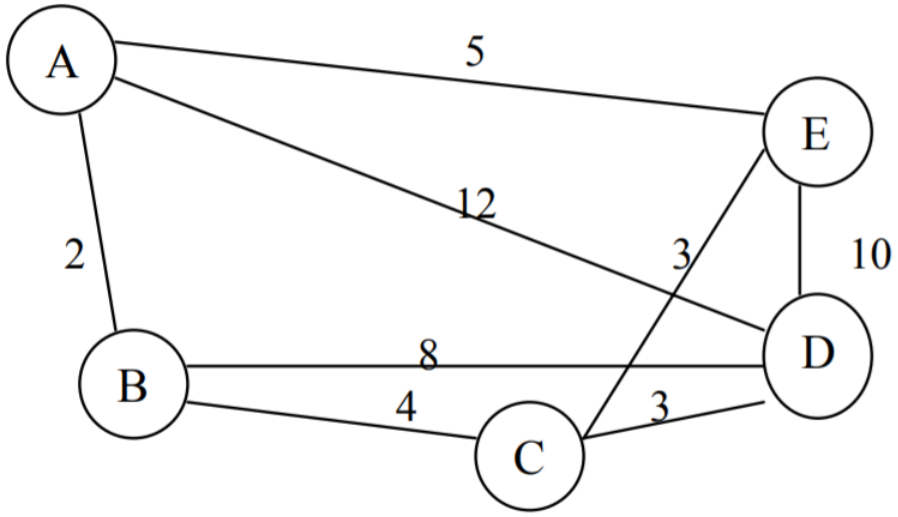

Figura 2. Exemple de TSP

De manera general, les instàncies de TSP venen definides per les següents dades:

- Dimensió:

$$n$$

- Conjunt de possibles solucions F:

$$f = n_1 - \cdots - n_{i=n}$$

- Funció objectiu:

$$c(f) = \sum_{i=2}^{n} \delta(n_i, n_{i-1})$$

Per tal d'il·lustrar el comportament del TSP amb dades numèriques, es farà servir l'exemple de la Figura 2. Trobem una instància amb les següents característiques:

- Dimensió:

$$n = 5.$$

- Conjunt de possibles solucions F:

$$f = n_1 - n_2 - n_3 - n_4 - n_5$$

Exemples:

$$f(1) = A - B - C - D - E$$

$$f(2) = B - D - A - E - C$$

$$f(3) = C - E - D - A - B$$

- Funció objectiu:

$$c(f(1)) = \delta(A,B) + \delta(B,C) + \delta(C,D) + \delta(D,E) = 2 + 4 + 3 + 10 = 19$$

$$c(f(2)) = \delta(B,D) + \delta(D,A) + \delta(A,E) + \delta(E,C) = 8 + 12 + 5 + 3 = 28$$

$$c(f(3)) = \delta(C,E) + \delta(E,D) + \delta(D,A) + \delta(A,B) = 3 + 10 + 12 + 2 = 27$$

Considerant el cas general del TSP, la quantitat de possibles solucions dins de F depèn directament de la dimensió n de la instància, augmentant de manera exponencial. Conseqüentment, aquest és considerat un problema del tipus NP-Hard [11].

2.1.2.2. Vehicle Routing Problem – VRP

El problema d'optimització de rutes es pot definir formalment de la següent manera [13]:

Un nombre n de clients amb una demanda coneguda d_i $(i = 1 \dots n)$ han de ser servits amb un nombre de vehicles m que surten d'un mateix magatzem. Aquests vehicles tenen una capacitat q per transportar béns. L'objectiu del problema és minimitzar el nombre de vehicles que es fan servir i la distància que recorre cada vehicle, de manera que els costos del servei siguin mínims.

El problema VRP presenta les següents restriccions:

1. La càrrega del vehicle no pot superar q.
2. Cada client és servit 1 sola vegada.
3. De manera necessària, totes les rutes tenen inici i final en un magatzem comú.

Matemàticament, el problema VRP es pot definir seguint les següents variables:

$$C = \{c_1, c_2, c_3 \dots c_n\} \rightarrow Clients$$

$$D = \{d_1, d_2, d_3 \dots d_n\} \rightarrow Demandes$$

$$V = \{v_1, v_2, v_3 \dots v_n\} \rightarrow Vehicles$$

$$q \rightarrow Capacitat$$

$$\gamma_{ij} \rightarrow Cost\ o\ distància\ entre\ els\ clients\ i, j$$

Un cop establertes les variables del problema, es pot definir la funció objectiu:

$$min \sum_{k=1}^{m} \sum_{i=0}^{n} \sum_{j=0}^{n} \gamma_{ij} \cdot x_{ijk}$$

Per altra banda trobem una sèrie de variables de decisió, que ens determinen la solució del problema:

- $x_{ijk} = \{0,1\}$: Variable binària que indica que el vehicle k arriba al client j provinent del client i.
- $y_{ki} = \{0,1\}$: Variable binària que indica que el vehicle k serveix al client i.

El conjunt de variables defineixen les següents restriccions al model:

$$\sum_{i=1}^{n} d_i y_{ki} \leq q, \quad k = 1,2 \dots m$$

$$\sum_{k=1}^{m} y_{ki} = 1, \quad i = 1,2 \dots n$$

$$\sum_{i=0}^{n} x_{ijk} = y_{kj}, \quad j = 0,1,2 \dots m, \quad \forall k$$

$$\sum_{j=0}^{n} x_{ijk} = y_{ki}, \quad i = 0,1,2 \dots n, \quad \forall k$$

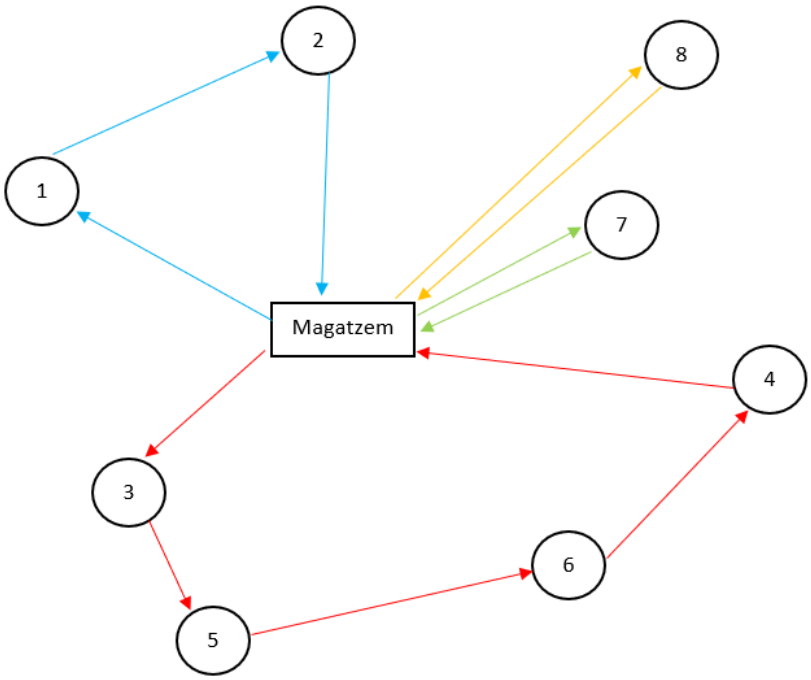

Figura 3. Exemple de problema VRP

En aquest cas, a diferència del problema TSP, trobem variables de decisió binàries. Seguint una mateixa estructura de problema, on es pot definir una instància d'entrada, una solució i un valor de funció objectiu associat a la solució, no necessàriament les dades que defineixen la estructura del problema segueixen un mateix patró.

Per altra banda, també es pot identificar un element nou, diferent del problema TSP: les restriccions. Afegir restriccions al problema augmenta la complexitat d'aquest i, per tant, el cost computacional de la seva resolució. El problema VRP és considerat un problema NP-Hard, ja que no s'ha trobat cap algoritme que aconsegueixi resoldre'l en temps polinòmic[13].

2.1.2.3. Knapsack Problem – KP

Finalment, es presenta un exemple de problema d'optimització combinatòria completament diferent als anteriors: El problema de la motxilla, conegut en anglès com a Knapsack Problem. De manera formal el problema es defineix de la següent manera[11]:

Donats el nombres no negatius següents:

$$n \rightarrow Nombre\ d'elements$$

$$c_i \rightarrow Valor\ dels\ elements$$

$$w_i \rightarrow Pes\ dels\ elements$$

$$W \rightarrow Capacitat\ màxima$$

S'ha de trobar un subconjunt $S \subseteq \{1, \dots, n\}$ que compleixi la següent condició:

$$\sum_{j \in S} w_j \leq W$$

La funció objectiu del problema KP ve definida per la següent expressió:

$$\sum_{j \in S} c_j$$

L'objectiu del problema és maximitzar la funció objectiu. De manera informal, es pot definir el problema KP com una motxilla on s'hi ha de ficar una sèrie d'objectes fins a omplir-la, de manera que el valor sumat de tots els objectes de la motxilla sigui el màxim possible. Novament, ens trobem amb un problema general que pot encaixar en multitud de problemes. Traslladant aquest concepte a aplicacions pràctiques dins del camp de l'enginyeria i la indústria, es pot recorre novament a una empresa de transport. En aquest cas, aquest model pot servir per definir quines comandes seran ateses per via aèria o per via terrestre. Les demandes més urgents (de més valor) seran ateses via aèria i la resta (les menys urgents) seran ateses per via terrestre. Com es pot intuir, la capacitat de l'avió és limitada i per tant s'ha de definir quines demandes seran ateses per aquesta via, de manera que es portin a terme el major nombre de comandes urgents possibles.

A continuació es presenta un exemple pràctic d'aplicació del problema KP. Donada una capacitat $W = 15$ la següent matriu de dades:

n	w	c
1	2	12
2	4	7
3	1	8
4	6	25
5	5	14

Taula 1. Exemple de problema KP

Algunes solucions al problema KP triades de manera aleatòria són:

$$f_1 \rightarrow 1-2-3-5$$

$$f_2 \rightarrow 1-2-3-4$$

$$f_3 \rightarrow 2-3-4-5$$

Els valors de la funció objectiu i del pes per a cada solució són:

$$c(f_1) = 12+7+8+14 = 41$$

$$w(f_1) = 2+4+1+5 = 12 \leq W$$

$$c(f_2) = 12+7+8+25 = 52$$

$$w(f_2) = 2+4+1+6 = 13 \leq W$$

$$c(f_3) = 7+8+25+14 = 54$$

$$w(f_3) = 4+1+6+5 = 16 \geq W \rightarrow No\ compleix$$

En aquest cas, la millor solució obtinguda és la f_2, amb un valor de la funció objectiu de 52. Com es pot comprovar, la solució f_3 proporciona un valor de funció objectiu millor però no compleix la condició que limita el pes total. Mitjançant un algoritme iteratiu, es poden obtenir millors solucions i per a instàncies de dimensions petites, aquest problema podria ser resolt en la seva totalitat. Tot i això, el problema KP mostra un increment exponencial de los possibles solucions a mesura que augmenta la dimensió n del problema, fent-lo intractable amb instàncies mitjanes i grans. Així doncs, aquest problema és considerat una vegada més com un problema NP-Hard[11].

Fins ara hem observat 3 casos diferents de problemes combinatoris, on hi han participat diferents tipus de variables, restriccions, solucions i funcions objectiu. En el següent apartat, tractarem en profunditat el problema d'optimització combinatòria que ens aborda en aquest estudi: El problema de flux de tasques o Flowshop Problem.

2.2. El problema del flux de tasques – "Flowshop Problem"

El problema a abordar en aquest estudi és anomenat generalment a la bibliografia com a "Flowshop Problem" (FP) o, alternativament per aquest cas concret, "Permutative Flowshop Problem" (PFP). El primer aborda el problema d'una manera més global, mentre que, d'altra banda, el PFP és un cas particular de FP.

El problema consisteix en una seqüència de tasques *n* que han de ser processades per un nombre de màquines *m*. Cada tasca n_j necessita d'un temps t_{ij} per a ser processada en la màquina m_i. L'objectiu és determinar quina seqüència de tasques permet optimitzar una funció objectiu. Aquesta funció objectiu, de manera general, és un paràmetre que ens permet avaluar el rendiment del procés productiu. Hi ha multitud de variables que poden fer-se servir com a funció objectiu per a avaluar la qualitat de la solució en el PFP. Entre aquestes variables de rendiment, la que es farà servir en aquest estudi és una de les més utilitzades a l'hora de resoldre el PFP. Es tracta del temps necessari per a processar la totalitat de les tasques a través de totes les màquines, és a dir, el temps que transcorre des de que es comença a processar la primera tasca per la primera màquina, fins que la última tasca acaba de processar-se per la última màquina. Aquest paràmetre és conegut pel seu nom en anglès "Makespan", el qual es farà servir d'ara en endavant en aquest estudi.

Formalment, la matriu de temps d'entrada del problema, on es defineixen els temps de procés de cada tasca a cada màquina és del tipus:

$$t_{ij} = \begin{pmatrix} t_{11} & \dots & t_{1n} \\ \dots & \dots & \dots \\ t_{m1} & \dots & t_{mn} \end{pmatrix}$$

La funció objectiu del problema és el makespan, de manera que es busca que aquest sigui mínim:

$$C_{min} = \sum_{j=1}^{n} \sum_{i=1}^{m} \max\{M_{i,j-1}; M_{i-1,j}\} + t_{ij}$$

On:

M_{ij}: Matriu de càlcul del makespan

t_{ij}: Matriu de temps d'entrada

Com s'ha comentat anteriorment, el PFP és un cas particular de FP. En el cas del FP, les tasques no necessàriament es processen sempre en el mateix ordre de màquines, de manera que el nombre de possibles solucions és $(n!)^m$. D'altra banda, al PFP totes les tasques es processen seguint la mateixa seqüència de màquines, de manera que el nombre de possibles solucions és $n!$. Per a contextualitzar la situació, a la Taula 2 s'exposa un petit exemple del que suposa aquesta simplificació.

		TASQUES		
		5	10	20
MÀQUINES	3	$1{,}728 \cdot 10^{6}$	$4{,}778 \cdot 10^{19}$	$1{,}440 \cdot 10^{55}$
	5	$2{,}488 \cdot 10^{10}$	$6{,}292 \cdot 10^{32}$	$8{,}524 \cdot 10^{91}$
	10	$6{,}192 \cdot 10^{20}$	$3{,}959 \cdot 10^{65}$	$7{,}265 \cdot 10^{183}$

Taula 2. Possibles solucions per a un exemplar de FP

		TASQUES		
		5	10	20
MÀQUINES	3	120	$3{,}629 \cdot 10^{6}$	$2{,}433 \cdot 10^{18}$
	5	120	$3{,}629 \cdot 10^{6}$	$2{,}433 \cdot 10^{18}$
	10	120	$3{,}629 \cdot 10^{6}$	$2{,}433 \cdot 10^{18}$

Taula 3. Possibles solucions per a un exemplar de PFP

Com es pot observar, aquesta simplificació no resulta trivial, ja que redueix el nombre de solucions significativament, la qual cosa ens permetrà assegurar amb una alta probabilitat trobar una solució de més qualitat, en un temps de càlcul millor. Tot i aquesta important reducció, per a instàncies grans del problema, es segueixen tenint un gran nombre de possibles solucions, de manera que l'ús d'algoritmes heurístics o metaheurístiques resulta completament necessari per a trobar una bona solució del problema.

A priori, la definició del problema sembla relativament senzilla. Ens trobem davant d'un problema combinatori on l'objectiu és optimitzar un paràmetre concret. L'abast d'aquest estudi però, va una mica més enllà amb l'objectiu de comparar el rendiment dels algoritmes existents enfront instàncies d'entrada estocàstiques. És a dir, els temps t_{ij} de cada tasca no són constants sinó que segueixen una distribució de probabilitat determinada prèviament. Per tant, resulta important diferenciar el PFP segons si la instància d'entrada segueix una matriu de temps determinista o estocàstica.

2.2.1. PFP amb temps deterministes

El model determinista del PFP és un dels problemes més estudiats durant dècades. Els primers treballs que van abordar-lo van basar-se en la resolució del problema per a instàncies de 2 màquines (Johnson, 1954). Aquest treball pioner va aconseguir establir una regla de resolució molt efectiva (regla de Johnson) que va permetre obrir el camí en aquest camp de treball per a instàncies més complexes. Tant és així que, durant la dècada dels 70's es van desenvolupar les

heurístiques CDS (Campbell, 1970), les quals permeten resoldre el problema per a instàncies superiors a 2 màquines. Bàsicament, el fonament dels algoritmes CDS és la reducció del problema de m màquines a un problema de 2 màquines, el qual es pugui resoldre mitjançant la regla de Johnson. Posteriorment es van començar a desenvolupar heurístiques més elaborades, com l'algoritme NEH (Nawaz, 1983), el qual es basa en la idea d'optimitzar el problema tasca a tasca, afegint-les a la solució progressivament.

Dins el camp de les metaheurístiques, s'han desenvolupat multitud d'algoritmes per a resoldre problemes d'optimització combinatòria, entre ells, el PFP. D'entre aquestes, destaquen el Simulated Annealing (Osman and Potts, 1989) – SA - o el Tabu Search (Widmer and Hertz, 1989) – TS - , millorat per Taillard (1990) i Reeves (1993). Entrant a un camp on la complexitat dels algoritmes augmenta, trobem els algoritmes genètics (Genetic Algorithms – GA) desenvolupats per Chen (1995), Reeves (1995), Wang i Zheng (2003) i Aldowaisan i Allahvedi (2003). En aquest camp d'investigació també destaquen els algoritmes de cerca local (Iterated Local Search – ILS) desenvolupats per Stützle (1998) i l'algoritme de la colònia de formigues (Ant Colony Optimization) desenvolupat per Chandrasekharan i Ziegler (2004). Finalment, cal destacar el desenvolupament d'algoritmes híbrids, els quals combinen diverses heurístiques i metaheurístiques anteriors. D'entre elles, destaca el GA combinat amb ILS o SA (Murata, 1996) [1]. Alguns dels algoritmes mencionats anteriorment seran descrits amb més detall posteriorment.

Per a resoldre el model determinista es tenen en compte les següents consideracions prèvies [2]:

1. Les màquines estan sempre disponibles.
2. Els temps de procés de cada tasca són deterministes i independents.
3. Els temps de setup de les màquines estan inclosos en els temps de procés.
4. Els temps de transport entre màquines són negligibles.
5. Una tasca no pot ser processada en més d'una màquina a la vegada.
6. Una màquina no pot processar més d'una tasca a la vegada.
7. Entre dues màquines les tasques poden esperar en un buffer de capacitat il·limitada.

2.2.1.1. Càlcul de la funció objectiu - El makespan

Com s'ha comentat amb anterioritat, hi ha diverses funcions objectius les quals es poden analitzar per tal de trobar una solució òptima. En aquest estudi, el paràmetre triat és un dels més usats en la literatura. Es tracta del makespan, també conegut en anglès com a "Completion time", el qual representa el temps transcorregut entre l'inici i el final del procés. Tal i com es podrà observar posteriorment, el càlcul d'aquest paràmetre suposa la clau en l'èxit de l'experiment computacional i conseqüentment, de l'estudi proposat. Els algoritmes metaheurístics tenen la capacitat d'analitzar gran diversitat de problemes, de manera que el seu funcionament roman pràcticament invariable de problema en problema, ja que, de manera general, el que fan és comparar paràmetres i escollir el millor segons un criteri definit prèviament. Així doncs, la clau de l'èxit està en el càlcul d'aquests paràmetres.

És clar doncs, que no suposa el mateix calcular el makespan per a una instància determinista i una instància amb dades estocàstiques. En una instància determinista, el que es pot trobar és una matriu de temps d'entrada com la de l'exemple il·lustrat en la Taula 4. El contingut de la matriu són els valors dels temps t_{ij} de cada tasca a cada màquina, els quals són constants i independents entre ells. És a dir, cada cop que l'algoritme sol·liciti calcular el makespan, en cada iteració, recorrerà a aquesta taula de temps.

A continuació, es detallarà quin és el procediment a seguir per a calcular el makespan en el model determinista. A mode d'exemple, es farà servir l'exemple de la Taula 4 on, com es determinarà al capítol posterior, la seqüència òptima és C-A-B-E-D. La taula de temps queda de la següent manera:

		TASQUES				
		C	A	B	E	D
MÀQUINES	1	1	4	10	9	7
	2	6	8	5	3	2

Taula 4. Matriu de temps d'entrada ordenats per a un PFP de 2 màquines i 5 tasques

Per a calcular el makespan en una instància de m màquines i n tasques, s'omple una matriu auxiliar M seguint els següents passos:

1. El valor de la cel·la $[1,1]$ de la matriu auxiliar M és igual al valor de la cel·la $[1,1]$ de la matriu de temps d'entrada t_{ij}.
2. La primera columna de la matriu auxiliar M s'omple seguint la següent equació:
$$M[i,1] = M[i-1,1] + t[i,j] \qquad (i \geq 2)$$
3. La primera fila de la matriu auxiliar M s'omple seguint la següent equació:
$$M[1,j] = M[1,j-1] + t[i,j] \qquad (j \geq 2)$$
4. La resta de cel·les de la matriu auxiliar M s'omplen seguint la següent equació:
$$M[i,j] = \max(M[i-1,j], M[i,j-1]) + t[i,j] \qquad (i \geq 2\ ;\ j \geq 2)$$
5. El makespan resultant és el valor de la cel·la $M[m,n]$.

En aquest cas, la matriu auxiliar M es construeix de la següent manera:

		TASQUES				
		C	A	B	E	D
MÀQUINES	1	1	5	15	24	31
	2	7	15	20	27	**33**

Taula 5. Matriu auxiliar M per a un PFP de 2 màquines i 5 tasques

Finalment, el makespan resultant és 33.

2.2.2. PFP amb temps estocàstics

A diferència del model determinista, el PFP amb temps estocàstics és un problema poc treballat a la literatura on pràcticament es desconeixen metodologies per abordar-lo de manera que es puguin obtenir resultats òptims i fiables. És per això que el desenvolupament d'aquest estudi és un repte, ja que la base teòrica actual és minsa. A més a més, aquesta base teòrica es centra en resoldre instàncies petites, de manera que per a les instàncies mitjanes i grans la base teòrica és pràcticament nul·la.

Si es posa l'atenció en els algoritmes existents per resoldre aquest problema, es pot observar que les alternatives són poques. Un dels pocs mètodes existents que ofereixen una bona solució al problema són les regles de Johnson i Talwar. Aquestes regles senzilles són útils per a resoldre instàncies per a 2 màquines i permeten obtenir resultats precisos i fiables. Si entrem dins del camp de les instàncies grans i mitjanes, les alternatives més utilitzades són els algoritmes CDS (Campbell, 1970) i NEH (Nawaz,1983). El funcionament és anàleg al de les heurístiques mencionades per al model determinista, i seran desenvolupades amb detall posteriorment. Cal remarcar un aspecte important respecte les heurístiques CDS. Aquest algoritme basa el seu funcionament en reduir els problemes grans en problemes de 2 màquines i un cop reduït, aquest es resol mitjançant les regles de Johnson i Talwar. Si bé la regla de Johnson és una eina més generalista, la regla de Talwar és efectiva davant de instàncies que treballen amb temps exponencials. Els algoritmes mencionats fins ara, juguen amb els paràmetres estadístics de les variables d'entrada, per generar una seqüència òptima del problema [4].

Per altra banda, trobem un altre camp de treball on existeixen multitud d'heurístiques i metaheurístiques, tot sovint anàlogues a les usades al model determinista, que s'apropen al camp de la simulació. És a dir, aquests algoritmes procedeixen de tal manera que basen part del seu funcionament en mètodes de simulació. Aquests mètodes es poden trobar a la literatura com a "Simheurístiques" [5] o "Models de simulació" [2], els quals també seran revisats amb detall en els següents capítols.

Per a resoldre el model estocàstic es tenen en compte les següents consideracions prèvies [2]:

1. Les màquines poden patir aturades.
2. Els temps de procés de cada tasca segueixen una distribució de probabilitat aleatòria i són independents.
3. Els temps de setup de les màquines estan inclosos en els temps de procés.
4. Els temps de transport entre màquines són negligibles.
5. Una tasca no pot ser processada en més d'una màquina a la vegada.
6. Una màquina no pot processar més d'una tasca a la vegada.
7. Entre dues màquines les tasques poden esperar en un buffer de capacitat il·limitada.

2.3. Heurístiques

2.3.1. Regla de Johnson

Aquest senzill algoritme permet calcular el makespan per a problemes amb instàncies d'entrada no superiors a 2 màquines. A continuació es mostra un exemple de PFP resolt mitjançant la regla de Johnson. Donada una matriu de temps d'entrada:

		TASQUES				
		A	B	C	D	E
MÀQUINES	1	4	10	1	7	9
	2	8	5	6	2	3

Taula 6. Matriu de temps d'entrada per a un PFP de 2 màquines i 5 tasques

La millor seqüència, amb la qual s'obté un makespan mínim es construeix seguint el següent algoritme:

1. Triar el temps t_{ij} de menor valor.
2. Si el temps seleccionat pertany a la màquina 1, la tasca associada a ell es situa a l'inici de la seqüència, en canvi, si pertany a la màquina 2, la tasca associada es situa al final de la seqüència.
3. Retorn al punt 1.

D'aquesta manera, en 5 iteracions, l'exemple proposat quedaria resolt de la següent manera:

C	-	-	-	-

C	-	-	-	D

C	-	-	E	D

C	A	-	E	D

C	A	B	E	D

Així doncs, la seqüència òptima amb la qual s'obté un makespan menor és C-A-B-E-D. Cal destacar que la regla de Johnson és aplicable tant al cas determinista com al cas estocàstic. En aquest segon però, només es recomanable fer-lo servir per a temps d'entrada que segueixin una distribució de probabilitat normal.

2.3.2. Regla de Talwar

El següent algoritme presenta altra vegada una formulació molt senzilla d'aplicar. Cal destacar, que en aquest cas, tractem un algoritme només aplicable per a resoldre el problema estocàstic amb temps d'entrada que segueixen una distribució de temps exponencial.

La matriu de temps d'entrada en el cas estocàstic, pot seguir qualsevol tipus de distribució de probabilitat. En el cas de la distribució exponencial, està definida pel paràmetre μ. Aquest paràmetre és el que es fa servir per a determinar una solució òptima al problema.

Per il·lustrar la regla de Talwar, definim els paràmetres μ, per a cada màquina i tasca:

		TASQUES				
		i_1	i_2	i_3	...	i_n
MÀQUINES	1	μ_{1i_1}	μ_{1i_2}	μ_{1i_3}	...	μ_{1i_n}
	2	μ_{2i_1}	μ_{2i_2}	μ_{2i_3}	...	μ_{2i_n}

Taula 7. Paràmetres μ per a cada màquina i tasca en un PFP estocàstic

Segons la regla de Talwar la tasca i_1 precedeix la tasca i_2 si es compleix la següent condició[2]:

$$\mu_{1i_1} + \mu_{2i_2} \geq \mu_{1i_2} + \mu_{2i_1}$$

De manera que es pot afirmar que si s'ordena la seqüència per ordre decreixent de $\mu_{i_1} - \mu_{i_2}$ aconseguirem minimitzar el valor del makespan esperat en un PFP estocàstic amb 2 màquines i temps d'entrada distribuïts exponencialment.

2.3.3. Heurística CDS

La heurística CDS (Campbell, 1970), sorgeix de la necessitat de resoldre el PFP per a valors de $m > 2$, ja que fins al moment només s'havia aconseguit resoldre el problema per a instàncies limitades a 2 màquines. Aquesta heurística brilla per la seva senzillesa i aplicabilitat en una rutina d'ordinador. A grans trets, la heurística CDS redueix una instància mitjana o gran a una instància de 2 màquines que es pot resoldre mitjançant la regla de Johnson o la regla de Talwar. La estratègia a seguir és la següent[2]:

1. Donada una matriu de temps d'entrada:

		TASQUES				
		i_1	i_2	i_3	...	i_n
MÀQUINES	m_1	t_{11}	t_{12}	t_{13}	...	t_{1n}
	m_2	t_{21}	t_{22}	t_{23}	...	t_{2n}
	...	...	...	...	...	...
	m_n	t_{m1}	t_{m2}	t_{m3}	...	t_{mn}

Taula 8. Matriu de temps d'entrada d'un PFP

S'agrupen els temps de procés en 2 màquines virtuals de la següent manera:

		TASQUES				
		i_1	i_2	i_3	...	i_n
MÀQUINES	m_{1-k}	$\sum_{i=1}^{k} t_{i1}$	$\sum_{i=1}^{k} t_{i2}$	$\sum_{i=1}^{k} t_{i3}$	...	$\sum_{i=1}^{k} t_{in}$
	m_{n-k+1}	$\sum_{i=k+1}^{n} t_{i1}$	$\sum_{i=k+1}^{n} t_{i2}$	$\sum_{i=k+1}^{n} t_{i3}$	...	$\sum_{i=k+1}^{n} t_{in}$

Taula 9. Matriu de temps d'entrada per a cada màquina virtual de CDS

Com es pot observar a la Taula 9, es poden formar $k = m - 1$ taules de temps. A partir d'aquí, es pot passar al següent pas.

2. Es calcula el makespan del problema reduït a 2 màquines virtuals per al valor de k actual, mitjançant la regla de Johnson o la regla de Talwar.

3. Es compara el makespan resultant per al valor k actual i es compara amb el millor makespan obtingut. Si el makespan actual és millor, el guardem i tornem al punt 2 per al següent valor de k.

A continuació es mostra el diagrama de flux de l'algoritme CDS.

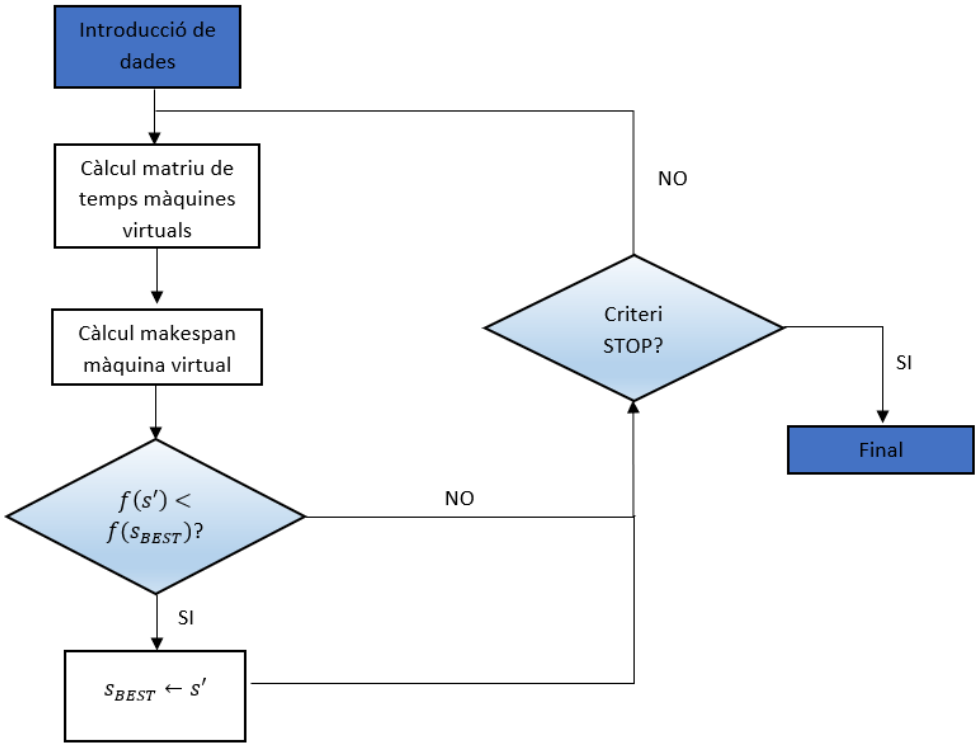

Figura 4. Diagrama de flux de la heurística CDS

A continuació es mostra un exemple de resolució d'una petita instància de PFP amb la heurística CDS. Donada una matriu de temps d'entrada:

		TASQUES				
		i_1	i_2	i_3	i_4	i_5
MÀQUINES	m_1	2	6	1	7	5
	m_2	1	2	5	3	3
	m_3	4	8	2	5	1

Taula 10. Matriu de temps d'entrada en un PFP amb 5 tasques i 3 màquines

Es calculen les matrius de temps de les màquines virtuals associades a cada valor de k:

$$k = 1$$

		TASQUES				
		i_1	i_2	i_3	i_4	i_5
MÀQUINES	m_{v1}	2	6	1	7	5
	m_{v2}	5	10	7	8	4

Taula 11. Matriu de temps de les màquines virtuals associades a k=1

$$k = 2$$

		TASQUES				
		i_1	i_2	i_3	i_4	i_5
MÀQUINES	m_{v1}	3	8	6	10	8
	m_{v2}	4	8	2	5	1

Taula 12. Matriu de temps de les màquines virtuals associades a k=2

Les solucions candidates per a cada valor de k calculat a través de la regla de Johnson, així com el seu valor de makespan associat són:

$$\boxed{f_{k=1} \rightarrow i_3 - i_1 - i_2 - i_4 - i_5 \rightarrow c(f_{k=1}) = 35}$$

$$f_{k=2} \rightarrow i_1 - i_2 - i_4 - i_3 - i_5 \rightarrow c(f_{k=2}) = 36$$

Finalment, la solució obtinguda a través de l'algoritme CDS és el corresponent a $k = 1$. De igual manera que passa amb l'algoritme NEH, que veurem al punt 2.3.1, la heurística CDS permet trobar ràpidament una solució, de manera que és una heurística útil per a trobar una "bona" solució inicial per a les metaheurístiques que es desenvoluparan durant l'estudi.

2.3.4. Heurística NEH

La heurística NEH (Nawaz, 1983) va ser desenvolupada com a continuació de les investigacions que van resultar en el desenvolupament de la heurística CDS, de manera que es pot dir que l'heurística NEH és fruit d'aquesta última heurística. El seu funcionament és molt senzill i es basa

en construir una solució tasca a tasca, començant amb una solució formada per dues tasques, on es van afegint les tasques restants una a una, agafant com a referencia el makespan parcial que resulta per a fer aquesta construcció. El procediment de la heurística NEH es resumeix en els següents 4 passos[14]:

1. Donada una matriu de temps d'entrada com la de la Taula 8, s'ordenen les tasques en ordre decreixent de temps de procés T_{ij}.

$$T_{ij} = \sum_{i=1}^{m} t_{ij}$$

$$Seqüència\ inicial\ \rightarrow T_{i1} - T_{i2} - T_{i3} - \cdots - T_{in}$$

2. Un cop ordenades les tasques en ordre decreixent de temps de procés, resulta una seqüència de tasques inicial. Seguidament, es seleccionen les dues primeres tasques $(k = 1\ i\ k = 2)$ d'aquesta seqüència i es calcula el makespan parcial:

   Primera solució parcial:

$$f_1 \rightarrow T_{i1} - T_{i2}$$

   Segona solució parcial:

$$f_2 \rightarrow T_{i2} - T_{i1}$$

   Makespan parcial:

$$\max\{c(f_1), c(f_2)\}$$

   Si el millor makespan parcial és $c(f_1)$ és seguirà construint la solució a partir de f_1. En cas contrari, es seguirà construint la solució a partir de f_2.

3. A partir de la tasca $(k = 3)$ es selecciona la següent tasca de la seqüència inicial calculada al pas 1 i s'insereix a la solució parcial actual. Depenent de la posició k que ocupi la tasca, hi haurà k possibles solucions noves, ja que l'algoritme NEH té com a condició que les solucions parcials generades han de respectar la seqüència de tasques entre elles. Si la solució actual és $f_1 \rightarrow T_{i1} - T_{i2}$, les possibles solucions noves inserint la tasca T_{i3} són:

$$f_1 \rightarrow T_{i3} - T_{i1} - T_{i2}$$

$$f_2 \rightarrow T_{i1} - T_{i3} - T_{i2}$$

$$f_3 \rightarrow T_{i1} - T_{i2} - T_{i3}$$

   Com es pot observar, l'ordre entre les tasques de la solució parcial anterior es manté invariable i s'ha inserit la nova tasca. Un cop construïdes les noves solucions candidates, es calcula de nou el makespan parcial per a cada una d'elles. La solució amb el millor makespan serà seleccionada per a seguir amb la heurística.

4. Si s'han esgotat totes les tasques de la seqüència inicial, l'algoritme ha finalitzat. En cas contrari, s'ha de tornar al punt 3.

A continuació es mostra el diagrama de flux de l'algoritme NEH.

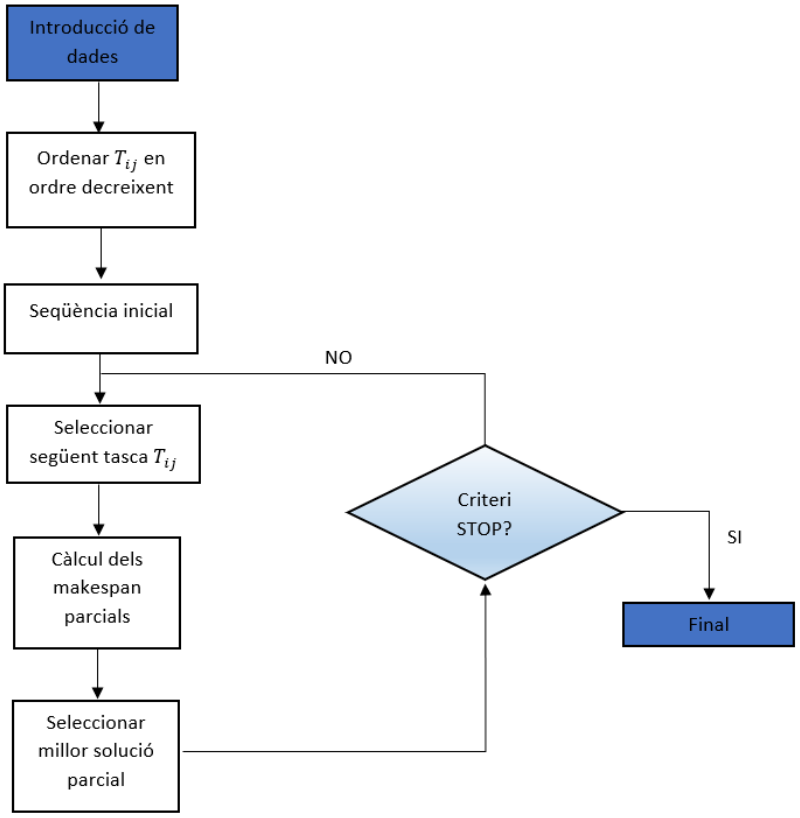

Figura 5. Diagrama de flux de la heurística NEH

A continuació es mostra un exemple de resolució d'una petita instància de PFP amb la heurística NEH.

Donada la matriu de temps d'entrada representada a la Taula 10, es vol calcular quina serà la seqüència òptima via la heurística NEH. Seguint els passos de la heurística, el primer pas a fer és ordenar les tasques per ordre decreixent del seu temps de procés:

		TASQUES				
		i_1	i_2	i_3	i_4	i_5
MÀQUINES	m_1	2	6	1	7	5
	m_2	1	2	5	3	3
	m_3	4	8	2	5	1
	T_i	7	16	8	15	9

Taula 13. Temps total de procés per tasca

De la Taula 10 s'extreu la següent seqüència inicial:

$$T_{i2} - T_{i4} - T_{i5} - T_{i3} - T_{i1}$$

Un cop obtinguda la seqüència inicial, procedim al pas 2. En aquest cas, seleccionem les 2 primeres tasques de la seqüència, que corresponen a $T_{i2} - T_{i4}$. Les 2 primeres solucions que sorgeixen són les següents:

$$\boxed{f_1 \rightarrow T_{i2} - T_{i4}}$$

$$f_2 \rightarrow T_{i4} - T_{i2}$$

El makespan parcial per a cada solució candidata és:

		TASQUES	
		i_2	i_4
MÀQUINES	m_1	6	7
	m_2	2	3
	m_3	8	5

Taula 14. Taula de temps parcial per a les tasques 2 i 4

$$c(f_1) = 21$$

		TASQUES	
		i_4	i_2
MÀQUINES	m_1	7	6
	m_2	3	2
	m_3	5	8

Taula 15. Taula de temps parcial per a les tasques 4 i 2

$$c(f_2) = 23$$

El resultat és que el makespan parcial òptim és el corresponent a la solució f_1, per tant, es seguirà construint la solució a partir d'aquesta solució parcial.

A continuació es procedeix amb el punt 3 de la heurística. La següent tasca a afegir a la solució és la tasca T_{i5}. Existeixen 3 possibles solucions parcials per a aquesta iteració:

$$f_1 \rightarrow T_{i5} - T_{i2} - T_{i4}$$

$$f_2 \rightarrow T_{i2} - T_{i5} - T_{i4}$$

$$\boxed{f_3 \rightarrow T_{i2} - T_{i4} - T_{i5}}$$

El makespan per a cada solució candidata és:

		TASQUES		
		i_5	i_2	i_4
MÀQUINES	m_1	5	6	7
	m_2	3	2	3

	m_3	1	8	5

Taula 16. Taula de temps parcial per a les tasques 5,2 i 4

$$c(f_1) = 26$$

		TASQUES		
		i_2	i_5	i_4
MÀQUINES	m_1	6	5	7
	m_2	2	3	3
	m_3	8	1	5

Taula 17. Taula de temps parcial per a les tasques 2,5 i 4

$$c(f_2) = 26$$

		TASQUES		
		i_2	i_4	i_5
MÀQUINES	m_1	6	7	5
	m_2	2	3	3
	m_3	8	5	1

Taula 18. Taula de temps parcial per a les tasques 2,4 i 5

$$c(f_3) = 22$$

Resulta doncs, que el millor makespan parcial és l'associat a la solució f_3, de manera que es seguirà construint la solució a partir d'aquesta solució parcial. A partir d'aquí, el punt 3 es va repetint fins esgotar les tasques de la seqüència inicial. La següent tasca a afegir és la tasca T_{i3}. Es plantegen les següents solucions candidates, juntament amb els seus valors de makespan associats:

$$\boxed{f_1 \rightarrow T_{i3} - T_{i2} - T_{i4} - T_{i5} \rightarrow c(f_1) = 23}$$

$$f_2 \rightarrow T_{i2} - T_{i3} - T_{i4} - T_{i5} \rightarrow c(f_2) = 24$$

$$f_3 \rightarrow T_{i2} - T_{i4} - T_{i3} - T_{i5} \rightarrow c(f_3) = 25$$

$$f_4 \rightarrow T_{i2} - T_{i4} - T_{i5} - T_{i3} \rightarrow c(f_4) = 28$$

El makespan parcial òptim sorgit correspon a la solució f_1, de manera que seguirem construint la solució a partir d'aquesta solució parcial. La següent tasca de la seqüència inicial és T_{i1}, de manera que les solucions candidates són:

$$\boxed{f_1 \rightarrow T_{i1} - T_{i3} - T_{i2} - T_{i4} - T_{i5} \rightarrow c(f_1) = 25}$$

$$f_2 \rightarrow T_{i3} - T_{i1} - T_{i2} - T_{i4} - T_{i5} \rightarrow c(f_2) = 26$$

$$f_3 \rightarrow T_{i3} - T_{i2} - T_{i1} - T_{i4} - T_{i5} \rightarrow c(f_3) = 27$$

$$f_4 \rightarrow T_{i3} - T_{i2} - T_{i4} - T_{i1} - T_{i5} \rightarrow c(f_4) = 27$$

$$f_5 \rightarrow T_{i3} - T_{i2} - T_{i4} - T_{i5} - T_{i1} \rightarrow c(f_5) = 27$$

Finalment, un cop inserides totes les tasques de la seqüència inicial, resulta la solució f_1 com la òptima, de manera que aquesta serà la solució final del problema. Com s'ha demostrat, la heurística NEH ens proporciona una bona solució al PFP de manera ràpida. Aquesta és una heurística que es pot implementar de manera senzilla en una rutina d'ordinador, de manera que, tal i com es veurà posteriorment, aquesta serà una de les heurístiques que es faran servir per a calcular una bona solució inicial en la metaheurística que s'implementarà en R, ja que introduir una solució inicial "bona" a l'algoritme, permetrà obtenir solucions de més qualitat.

2.3.5. Algoritmes de cerca local

A continuació i durant els propers apartats, es definiran les metaheurístiques que es fan servir durant l'estudi. A diferència de les heurístiques anteriors, aquest tipus d'algoritmes permeten trobar solucions per a multitud de problemes diferents. En canvi, les heurístiques que s'han vist fins al moment permetien resoldre un problema concret, en aquest cas, el PFP. Aprofundint en el camp de les metaheurístiques, els algoritmes de Simulated Annealing i de Tabu Search són anomenats algoritmes de cerca local, ja que permeten optimitzar una solució inicial mitjançant petites modificacions en les solucions candidates, de manera que les solucions que van sorgint durant l'execució de l'algoritme resulten pertànyer a un mínim o màxim local.

Per entendre millor els algoritmes de cerca local, es mostra gràficament a la Figura 6 el funcionament d'aquest tipus d'algoritmes.

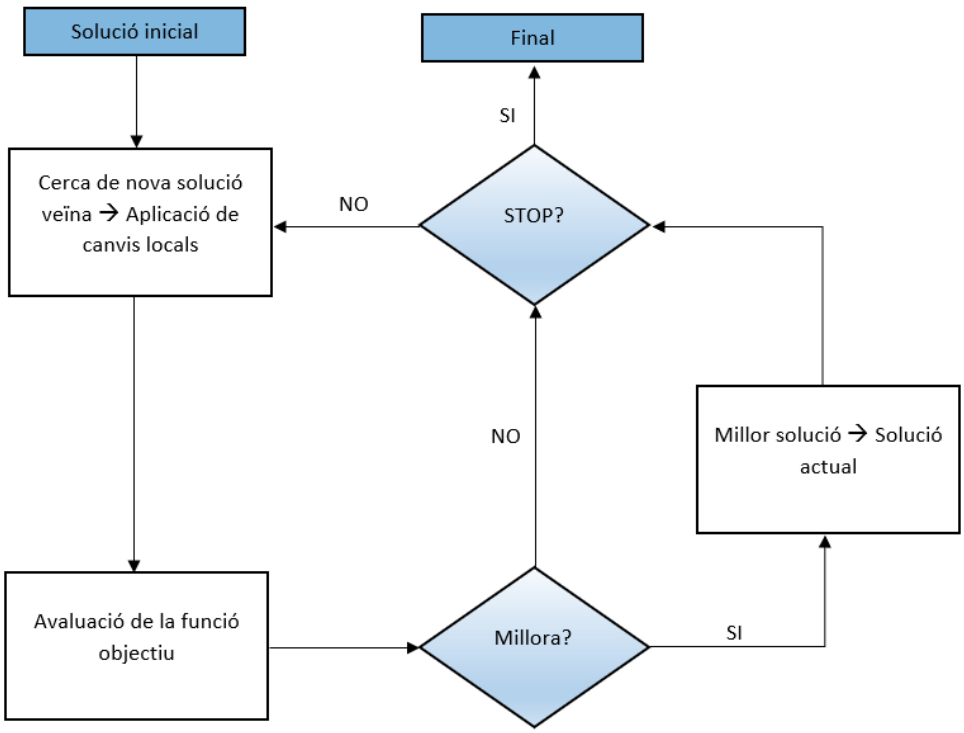

Figura 6. Diagrama de flux d'un algoritme de cerca local

2.3.5.1. Simulated Annealing (SA)

La heurística SA té el seu origen en la indústria metal·lúrgica i traduït literalment significa recuit simulat. El nom de la metaheurística no és trivial, ja que el comportament d'aquest algoritme simula la disminució de temperatura d'un metall a mesura que passa el temps.

S'ha de posar especial atenció al punt del diagrama de la Figura 6 on es cerca una nova solució. En aquest punt de l'algoritme es cerca una solució entre totes les solucions veïnes. És el concepte conegut en anglès com "neighbourhood", el qual vol dir fer una petita modificació a la solució actual per obtenir una nova solució similar a la actual, la qual pot resultar en una millora de la funció objectiu. En general, sol ser un intercanvi (swap) de 2 nodes en la seqüència solució actual (TSP i VRP) o, en el cas del PFP, un intercanvi de 2 nodes en la seqüència de tasques actual. A les Figures 7 i 8 es poden observar dos exemples de canvis locals[15].

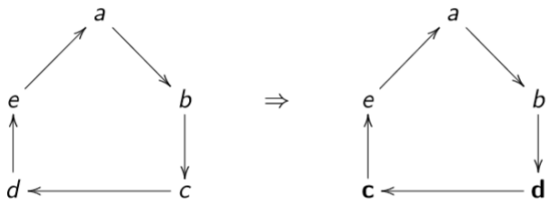

Figura 7. Canvi local en un problema TSP

1 **21**345
2 1**32**45
3 12**43**5
4 123**54**

Figura 8. Canvi local en un PFP per una solució inicial 12345

Per altra banda, cal destacar una important característica de l'algoritme SA. Aquesta és la possibilitat que ofereix l'algoritme d'acceptar una solució pitjor que la solució actual. Aquesta opció s'activarà en funció d'una corba exponencial que depèn del nombre d'iteracions executades per l'algoritme en aquell moment i d'un paràmetre μ que permet regular el comportament d'aquesta característica. Així doncs, l'algoritme SA ens permet acceptar solucions dolentes amb una probabilitat més elevada a l'inici de l'algoritme, on trobem les pitjors solucions i amb una probabilitat més baixa a mesura que l'algoritme va avançant i obtenim millors solucions. A la Figura 9 [15] es pot observar quina és la probabilitat de quedar-se una solució pitjor a la actual en funció de les iteracions executades i per diferents valors del paràmetre μ.

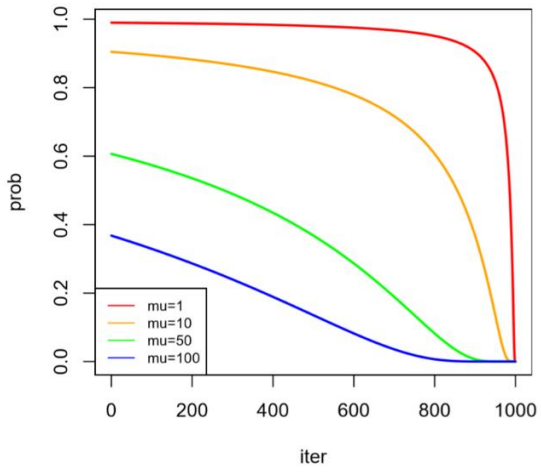

Figura 9. Probabilitat d'acceptar una mala solució en funció de la iteració i μ

Per tal d'obtenir un bon rendiment de l'algoritme SA, s'han de sintonitzar correctament els dos paràmetres que defineixen la corba de la Figura 9. Per una banda, un valor molt baix de la relació de paràmetres $\frac{\mu}{T}$ implica una alta probabilitat d'acceptar una mala solució, el que vol dir que al final de l'algoritme obtindrem una solució dolenta. Per altra banda, un valor molt alt d'aquest quocient de paràmetres implica que la probabilitat d'acceptar una mala solució sigui molt baixa, de manera que es corre el risc d'estancar-se en un mínim o un màxim local. Així doncs, l'objectiu de la sintonització d'aquests paràmetres és obtenir un quocient equilibrat, similar al que es pot observar a la corba blava de la Figura 9, que permeti acceptar una mala solució al principi de l'execució de l'algoritme, però que aquesta probabilitat vagi disminuint gradualment a mesura que les iteracions executades augmenten, de manera que al final de l'algoritme, on tenim les millors solucions, la probabilitat de quedar-se la solució bona sigui molt elevada.

A continuació es mostra el diagrama de flux per a l'algoritme SA.

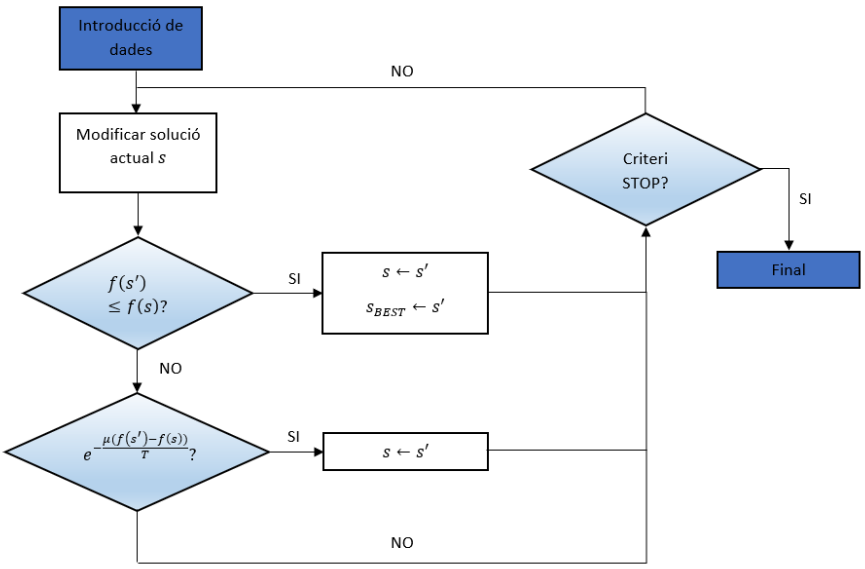

Figura 10. Diagrama de flux de la metaheurística SA

2.3.5.2. Tabu Search (TS)

Continuant en el camp dels algoritmes de cerca local, trobem la metaheurística Tabu Search (Cerca Tabú). De la mateixa manera que en el cas del SA, aquest algoritme és aplicable a multitud de problemes ja que segueix una estructura idèntica a la que es mostra a la Figura 6. El seu funcionament és novament senzill i fàcilment reproduïble en un programa informàtic. Tal i com indica el seu nom, aquesta metaheurística basa el seu funcionament en vetar certs moviments per a aconseguir que l'algoritme avanci en la seva cerca d'una solució òptima.

Novament, de igual manera que passava amb l'algoritme SA, aquest punt característic ve definit a l'hora de buscar una nova solució d'entre les solucions veïnes. El funcionament que fa servir l'algoritme TS per trobar una nova solució és el següent:

1. Donada una solució actual s, es vol seleccionar una nova solució d'entre les solucions veïnes. El mecanisme per a modificar la solució actual és el mateix seguit a l'algoritme SA, consistent en intercanviar la posició de 2 nodes. Aleshores, per a una instància de dimensió n, tenim $n-1$ solucions veïnes, tal com s'il·lustra a la Figura 8. A aquest conjunt de solucions veïnes se l'anomena $N(s)$.
2. Un cop s'han generat les solucions veïnes $N(s)$, es calcula el valor de funció objectiu per a cada una d'elles. En el cas del PFP, es tractarà de calcular el valor de makespan per a cada solució veïna.
3. D'entre les solucions de $N(s)$, escollirem la millor d'elles, sigui o no millor que la solució actual. Aquest punt és diferencial respecte l'algoritme SA, ja que en aquest últim, només es seleccionen les solucions millors que la solució actual i les pitjors amb una probabilitat $e^{-\frac{\mu(f(s')-f(s))}{T}}$, depenent del valor de T. En el cas de TS, si la solució trobada és la millor trobada durant la cerca, es guarda igualment com a millor solució i es segueix iterant.
4. Per últim, s'actualitza la llista tabú. Aquest pas consisteix en guardar l'últim moviment realitzat per trobar la solució actual, de manera que aquest moviment queda vetat durant les properes iteracions.

A continuació s'il·lustra el concepte de llista tabú.

$$Solució\ actual \rightarrow 12345$$

$$Nova\ solució \rightarrow 12\mathbf{43}5$$

Donada una solució actual, la nova solució veïna s'actualitza a solució actual i es guarda l'intercanvi entre els nodes 3 i 4 com a moviment tabú a la llista. D'aquesta manera, durant les properes iteracions no s'acceptarà una solució generada realitzant aquest intercanvi de nodes. Aquest paràmetre ve definit per l'usuari. Aquest simple criteri, és clau per fer avançar l'algoritme. Així doncs, a l'acceptar la millor solució de les solucions veïnes, si també contempléssim aquests moviments vetats, l'algoritme entraria fàcilment en un bucle entre 2 solucions que el faria encallar-se. A continuació s'il·lustra aquest possible problema.

Tornant a l'exemple anterior, simularem uns cicles de l'algoritme de cerca tabú. La solució inicial és $f_0 = 12345$. A la següent iteració es generen totes les solucions veïnes i sorgeixen les següents solucions candidates, amb els seus respectius valors de la funció objectiu:

$$Solució\ actual\ f_0 \rightarrow 12345\,; c(f_0) = 11$$

$$f_1 \rightarrow \mathbf{21}345\,; c(f_1) = 12$$

$$f_2 \rightarrow 1\mathbf{32}45\,;\ c(f_2) = 13$$

$$\boxed{f_3 \rightarrow 12\mathbf{43}5\,;\ c(f_3) = 10}$$

$$f_4 \rightarrow 123\mathbf{54}\,;\ c(4) = 11$$

La solució veïna escollida és la f_3, ja que presenta un valor de funció objectiu mínim. Així, doncs aquesta solució es converteix en la solució actual. Si no es restringís l'intercanvi de nodes 3-4 en una llista tabú, a la següent iteració podria ocórrer la següent casuística:

$$Solució\ actual \rightarrow 12435$$

$$f_1 \rightarrow \mathbf{21}435\,; c(f_1) = 14$$

$$f_2 \rightarrow 1\mathbf{42}35\,;\ c(f_2) = 13$$

$$\boxed{f_3 \rightarrow 12\mathbf{34}5\,;\ c(f_3) = 11}$$

$$f_4 \rightarrow 124\mathbf{53}\,;\ c(4) = 12$$

En aquesta iteració, la solució veïna que resulta ser la millor és la f_3, que resulta ser la solució de partida de la iteració anterior. Si això es permet, en la següent iteració l'algoritme tornaria a seleccionar com a millor solució veïna la solució 12435 i hauria quedat encallat en un bucle que no interessa per a que pugui avançar. D'aquesta manera, queda demostrada la necessitat de restringir els últims moviments per tal de fer avançar l'algoritme cap a una solució òptima.

A continuació es mostra el diagrama de flux de l'algoritme TS.

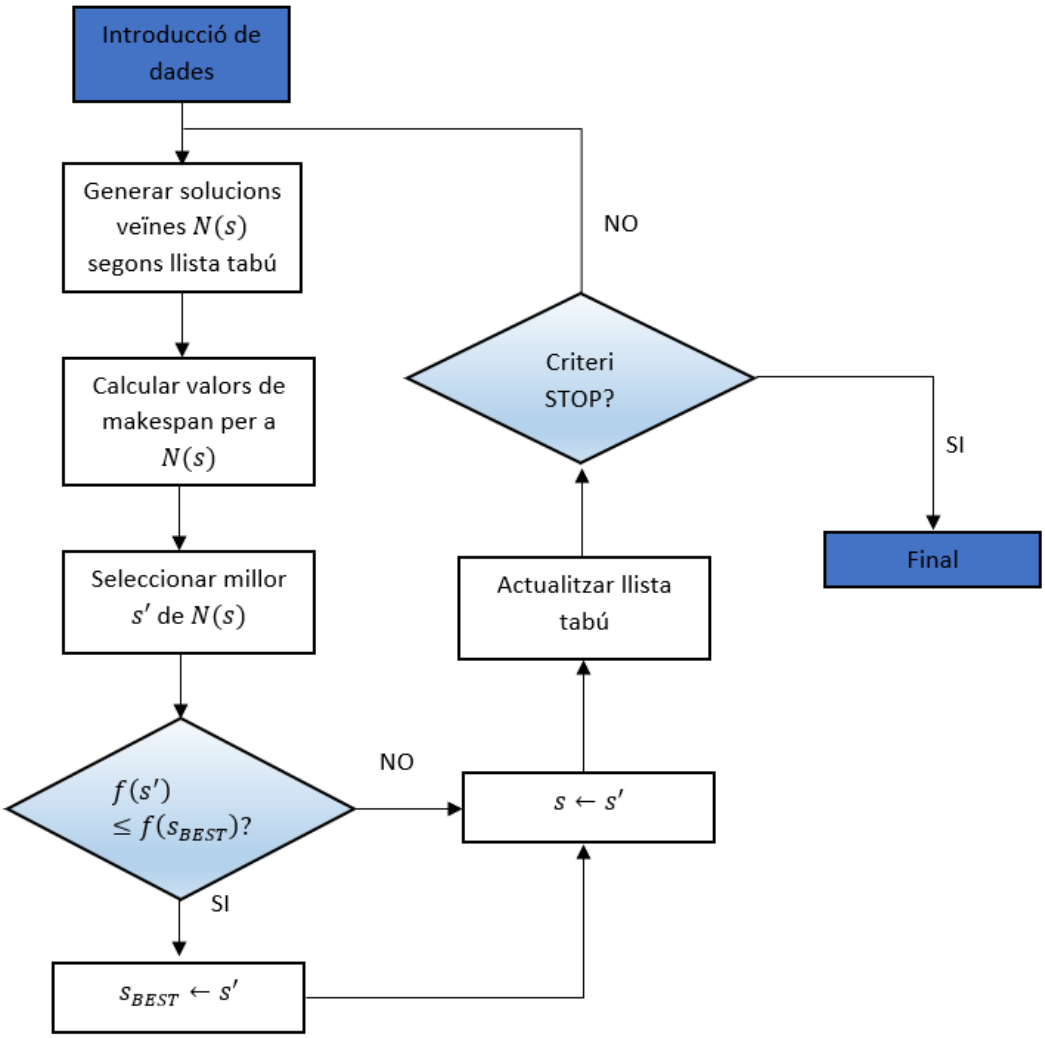

Figura 11. Diagrama de flux de la metaheurística TS

2.3.7. Iterated Local Search (ILS)

Fins al moment s'han definit dues metaheurístiques que permetien cercar solucions en el marc d'un màxim o mínim local. A continuació es pretén anar més enllà i no limitar la cerca a màxims i mínims locals, de manera que es puguin explorar solucions en un rang molt més ampli. Si representem gràficament les possibles solucions al problema d'optimització combinatòria contraposant-les als seus respectius valors de funció objectiu, obtindrem un gràfic com el de la Figura 12.

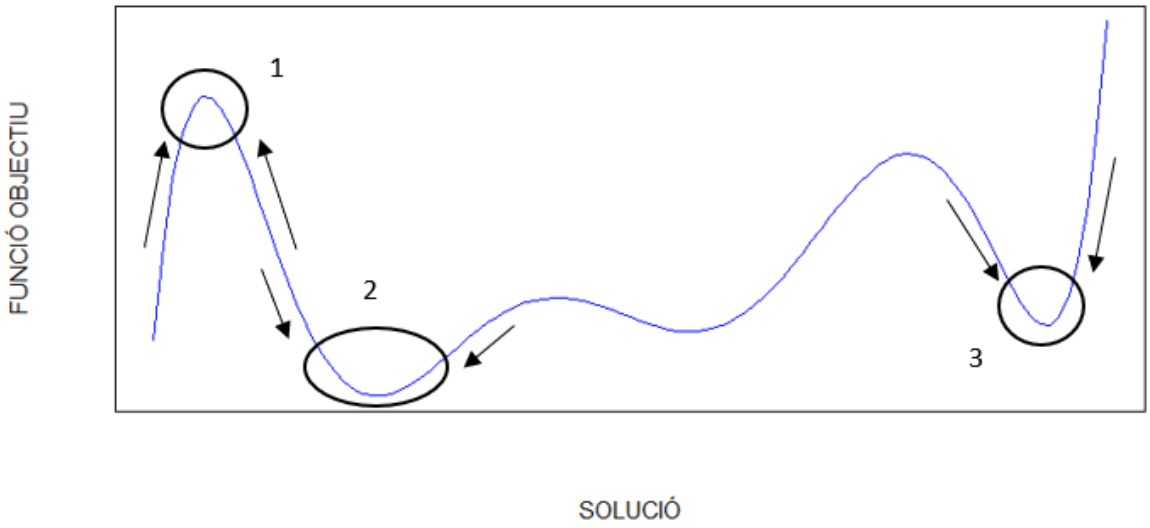

Figura 12. Evolució de la funció objectiu en funció de la solució obtinguda

Els algoritmes de SA i TS permeten ascendir o decréixer en un mínim o màxim local ja que la cerca de solucions candidates es basa en l'intercanvi de dos nodes. Aquesta és una petita modificació en la solució actual que permet obtenir una solució candidata molt propera a la actual i no permet escapar de l'òptim local.

Per contraposició, els algoritmes de cerca local iterada tenen per objectiu escapar d'aquests òptims locals i expandir l'exploració de solucions més enllà d'aquests. Si agafem com exemple el gràfic de la Figura 12, un algoritme ILS permet fer un salt des del punt 3 fins al punt 2. Aquest salt en la funció implica un canvi substancial de la solució actual per tal d'explorar una nova zona del gràfic. Aquest canvi s'anomena pertorbació de la solució actual.

Els elements que conformen un algoritme ILS són:

1. Una heurística de cerca local. Aquest element permet cercar solucions en el marc d'un òptim local.
2. Mecanisme de pertorbació de la solució. Aquest element ha de permetre modificar la solució actual de manera que permeti escapar l'algoritme de màxims i mínims locals, en busca del màxim o mínim global. Hi ha multitud de mecanismes per pertorbar la solució actual, però tots han de complir unes condicions. El mecanisme de pertorbació ha de ser suficientment fort per a permetre escapar de l'òptim local, però no suficientment fort com per provocar un reinici de l'algoritme. És a dir, ha de permetre fer el salt d'un òptim local a un altre contigu o proper. El mètode de pertorbació que s'utilitza en aquest estudi s'anomena "Shift to left operator" i consisteix en els següents passos:
    a. Seleccionar dos nodes aleatoris i intercanviar les seves posicions.
    b. Seleccionar un dels nodes anteriors i intercanviar la seva posició amb el node de la seva esquerra.
    c. Calcular el makespan de la nova seqüència generada.
    d. Repetir la operació descrita al punt b i c fins que el node seleccionat al punt a sigui el primer node de la seqüència.

e. Un cop s'han esgotat les possibilitats amb el primer node, repetir els punts b, c i d amb el segon node seleccionat al punt a.
f. La solució candidata per continuar amb l'algoritme serà la que tingui un millor valor de makespan associat.

3. Criteri d'acceptació. L'algoritme ILS ha de tenir una manera de valorar si una solució candidata s'accepta o no. Generalment, aquest criteri es basa en la avaluació de la funció objectiu.

A la Figura 9 es mostra el diagrama de flux que segueix un algoritme ILS.

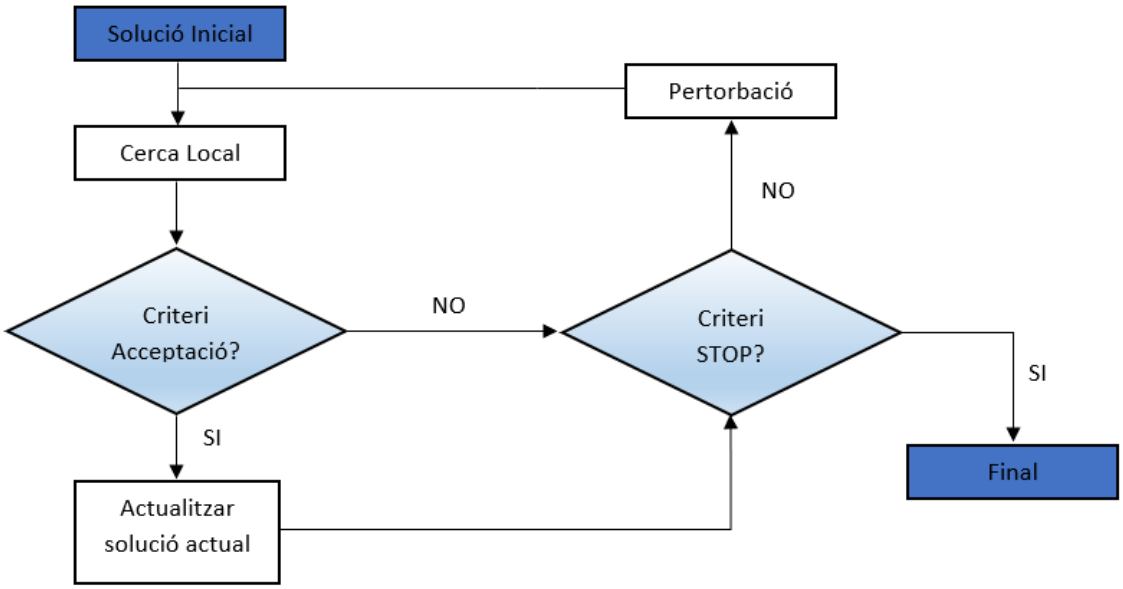

Figura 13. Diagrama de flux d'un algoritme ILS

2.3.8. Simulació + ILS (SIMILS)

L'últim pas per a completar el marc teòric de l'estudi és desenvolupar un algoritme ILS que s'adapti a les necessitats del PFP amb temps estocàstics. Aquest factor, obliga a afegir un component de simulació a l'algoritme ILS per tal de poder treballar amb dades que no presenten un comportament determinista, sinó que segueixen una distribució de probabilitat determinada.

L'elecció d'una bona estratègia per un algoritme que treballa amb dades no deterministes és clau per a l'execució del programa en R. Una estratègia inicial pot ser la de realitzar una simulació del valor de la funció objectiu cada cop que el programa sol·liciti el seu càlcul. Aquesta estratègia pot presentar una alta fiabilitat de les solucions, però representa un consum de temps de computació molt elevat, de manera que no resulta una estratègia factible. La estratègia escollida doncs, passa per a adaptar l'algoritme ILS a PFP amb temps deterministes a un algoritme que treballa amb solucions deterministes i avalua la funció objectiu mitjançant tècniques estadístiques de simulació per a les solucions prometedores. La simulació del valor de la funció

objectiu per a una solució candidata requereix d'un important esforç computacional, de manera que realitzant aquesta simulació únicament per a les solucions prometedores s'aconsegueix un estalvi del temps de computació notable. A la Figura 14 es mostra el diagrama de flux de l'estratègia seguida per a l'algoritme de Simulació + ILS, el qual anomenarem d'ara en endavant com a SIMILS[17].

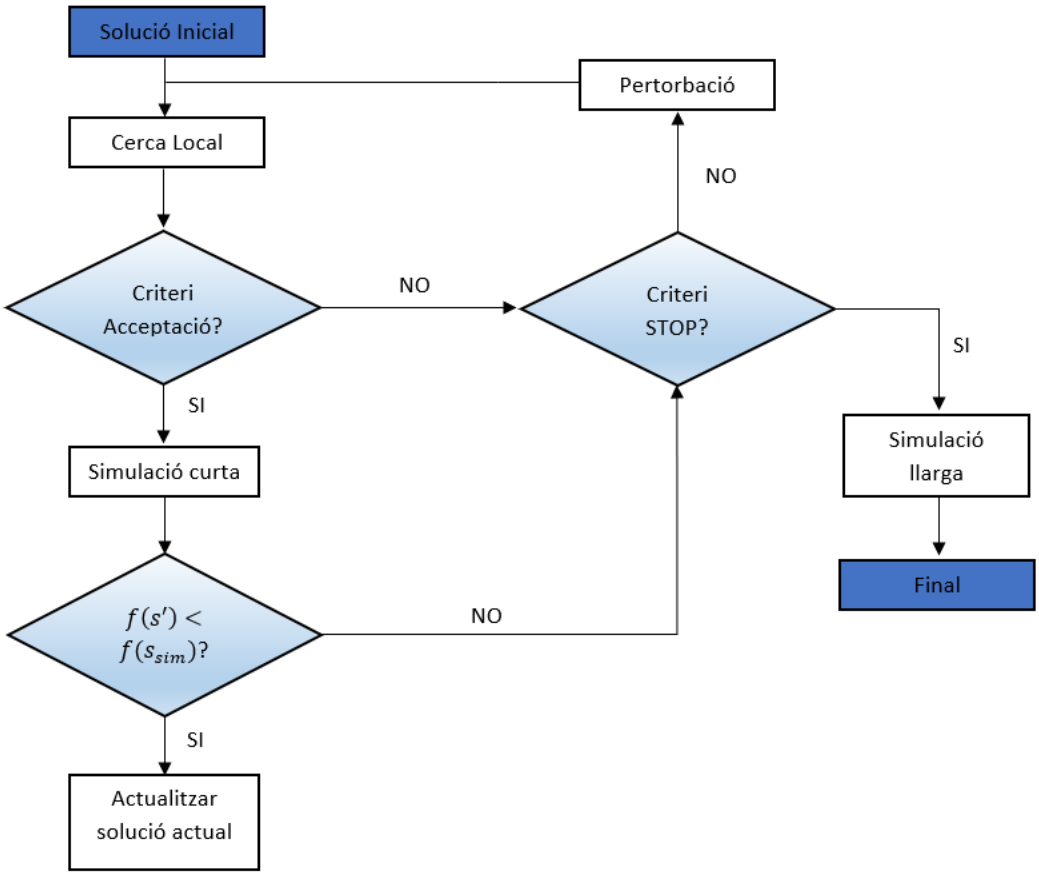

Figura 14. Diagrama de flux de l'algorisme SIMILS

Com es pot observar a la Figura 14, l'algoritme SIMILS procedeix amb una solució inicial determinista i opera amb una heurística de cerca local determinista. Un cop es troba una solució determinista prometedora, es procedeix a fer una simulació curta del seu valor associat de funció objectiu. A aquest valor, per al problema que ens incumbeix i tal i com es podrà veure amb posterioritat, se l'anomenarà EMS, de les sigles en anglès "Expected Makespan". La seva traducció al català és Makespan esperat, el qual representa l'esperança matemàtica del makespan simulat. Cal destacar que la simulació que es realitza per a cada solució prometedora és una simulació curta, que té per objectiu determinar un valor de EMS fiable per a seguir amb l'algoritme d'una manera ràpida. Finalment, un cop l'algoritme ha finalitzat, es realitza una simulació llarga per obtenir un resultat de EMS amb una alta fiabilitat i poder analitzar-ne els seus paràmetres estadístics[16].

3. Desenvolupament d'heurístiques mitjançant "R"

3.1. Introducció

Arribats a aquest punt de l'estudi, és convenient fer balanç del tot allò que s'ha vist fins ara i tot allò que encara queda per veure. Tal i com s'ha definit a l'inici del present estudi, aquest té com a objectiu comparar diverses heurístiques que resolguin el problema de flux de tasques, de manera que es pugui avaluar quina d'elles resulta ser la més indicada en termes de qualitat de la solució i d'eficiència computacional per a resoldre'l. Fins al moment, s'han definit què són els problemes d'optimització combinatòria de manera genèrica i sobre què tracta el problema del flux de tasques de manera més concreta. Per altra banda, també s'han definit els conceptes claus de diferents heurístiques i metaheurístiques.

És en aquest moment que, després d'haver assentat les bases del marc teòric en que es basa l'estudi, aquest passa a la seva part més pràctica. Durant el present apartat es desenvoluparan les heurístiques i metaheurístiques que s'han vist fins ara en un codi d'ordinador, el qual permeti resoldre el problema de flux de tasques per a diverses instàncies d'entrada i avaluar-ne els resultats, així com diferents paràmetres estadístics que poden resultar d'interès per a comparar els diferents algoritmes.

3.2. Package Rcpp

El llenguatge de programació usat per a aquest estudi és el R. Si bé aquest llenguatge és més "amigable" per a l'usuari que d'altres llenguatges, també és més lent executant-se que d'altres llenguatges. És per això que s'ha decidit fer servir el package Rcpp per tal de millorar el rendiment de l'execució del programa. A grans trets, aquest paquet de codi permet "traduir" automàticament codi C++ a R, de manera que es poden desenvolupar funcions en codi C++ les quals poden integrar-se a un programa en R sense problemes. A ulls de l'usuari, quan es crida una funció en C++ aquesta s'executa com si es tractés d'una funció en R, però la realitat és que la funció s'està executant en C++. Aquesta eina doncs, resulta ser de gran utilitat per a millorar el rendiment del codi en termes de temps de d'execució, ja que el llenguatge C++ és substancialment més ràpid que el codi en R. A la Figura 15[18] es mostra una comparativa, en termes de velocitat ,entre diferents llenguatges de programació per a resoldre diferents problemes.

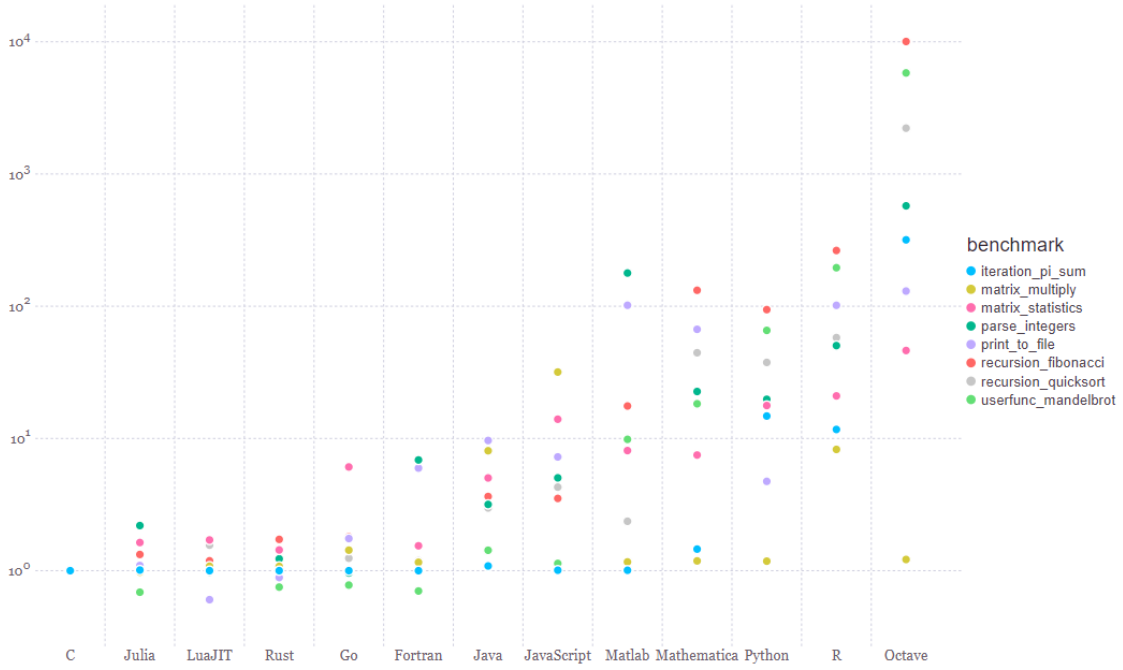

Figura 15. Comparació de llenguatges de programació en termes de velocitat d'execució

En el cas que s'està estudiant, es desenvoluparan les funcions claus en C++, mitjançant el package Rcpp, però no la totalitat del programa, que es desenvoluparà en R. Les funcions que s'han escollit per a desenvolupar en C++ són:

- makespanC: Funció per a calcular el valor de makespan
- TimeMatrixGenerator: Funció que genera una matriu de temps aleatòria en base a una distribució de probabilitat.

Aquestes funcions són les més recursives durant l'execució del programa principal, de manera que fent servir Rcpp per a aquestes 2 funcions serà suficient per a millorar el rendiment del programa principal de manera substancial. Per quantificar quina és la millora pel que fa a temps d'execució a l'hora de fer servir Rcpp, s'han testejat aquestes funcions en R i C++ per comparar els temps d'execució. La llibreria de R "microbenchmark" permet executar ambdós programes i comparar el temps que triguen en executar-se.

La ordre "microbenchmark" executa un nombre de repeticions determinat per l'usuari, que en aquest cas s'ha establert en 100, i ofereix dades estadístiques sobre el temps d'execució de les dues funcions que s'han introduït per analitzar. Un cop executats, els resultats són els següents:

```
Unit: microseconds
                          expr     min      lq     mean median     uq    max neval
 MakespanDet(tai205$tij, 1:20) 34.441 35.470 38.12252 35.984 37.012 71.454   100
   makespanC(tai205$tij, 1:20)  5.140  5.655  6.89865  6.169  6.683 34.442   100
```

Figura 16. Resultats de microbenchmark per a la funció makespan

Com es pot observar a la Figura 16 el valor mitjà del temps d'execució per a la funció MakespanDet (R) és de 38,12 µs. Per altra banda el valor mitjà del temps d'execució per a la funció makespanC (C++) és de 6,89 µs. Per tant, en aquest cas el codi de C++ s'executa de l'ordre

d'un 550% més ràpid que el codi en R. Aparentment, pot semblar que accelerar una funció que té un temps d'execució de microsegons no té perquè influir en el resultat final. Però si es calcula el nombre de vegades que aquesta funció és cridada al programa principal, es pot comprovar que no és trivial deixar de banda aquesta millora. En el cas de la instància més petita que es simula en aquest estudi, es recorre a la funció de càlcul de makespan fins a 7220 vegades cada cop que es realitza una cerca local. A la vegada, aquesta cerca local es realitza un nombre de vegades $TREPBase = 1000$. Per tant, la funció és executada fins a 7.220.000 vegades. Queda clar doncs, que aquesta millora influeix de manera apreciable en el temps global d'execució del programa principal.

Per altra banda, els resultats obtinguts per a la funció TimeMatrixGenerator es poden observar a la Figura 17.

```
Unit: microseconds
                                                expr     min      lq      mean  median      uq     max neval
 TimeMatrixGenerator(tai205$tij, VarMat205, 1:20) 153.189 155.245 166.17906 157.301 163.4695 262.169   100
      TimeMatrixGeneratorC(tai205$tij, VarMat205)   7.711   8.225   9.90078   9.253  9.7670  30.329   100
```

Figura 17. Resultats de microbenchmark per a la funció TimeMatrixGenerator

Com es pot observar, la millora en el cas de la funció TimeMatrixGenerator és del 1700%.

3.3. Funcions

A continuació es desenvolupa el codi per a totes les funcions necessàries per a resoldre el problema del flux de tasques mitjançant diferents heurístiques. El codi de totes les funcions desenvolupades es pot consultar a l'Annex adjunt a la present memòria.

3.3.1. makespanC

La funció makespanC és una de les funcions bàsiques del programa principal, ja que es fa servir en absolutament totes les heurístiques. La seva funció és, donades una matriu de temps d'entrada i una seqüència solució en forma de vector, la de calcular el valor de la funció objectiu, que tal com és sabut, per al problema que ens aborda és el makespan. Aquesta és una de les funcions dins del codi desenvolupat a l'estudi, que s'ha programat en llenguatge C++ i encaixada al codi global en R mitjançant el paquet Rcpp per a accelerar la velocitat de computació de programa.

El codi associat a aquesta funció és:

Variables d'entrada:

- Matriu d'enters → Matriu de temps t
- Vector d'enters → Vector solució s

Variables de sortida:

- Vector d'enters → Solució obtinguda z

$int\ \boldsymbol{makespanC}(NumericMatrix\ t, NumericVector\ s)\{$

$\#\boldsymbol{Declaració\ de\ variables}:$

$int\ m = t.nrow(\,), n = t.ncol(\,);$

$NumericMatrix\ u(m, n);$

$NumericVector\ p;$

$int\ i, j;$

$p = s - 1;$

$int\ z;$

$u(0,0) = t(0, p[0]);$

$\#\boldsymbol{Càlcul\ del\ makespan\ mitjançant\ matriu\ auxiliar\ u}$

$\boldsymbol{for}(i = 1; i < m; i++)\ \{u(i, 0) = u(i - 1,0) + t(i, p[0]);\}$

$\boldsymbol{for}(j = 1; j < n;\ j++)\ \{$

$u(0, j) = u(0, j - 1) + t(0, p[j]);$

$\boldsymbol{for}(i = 1; i < m; i++)\{$

$u(i, j) = \max\Big(NumericVector :: create\big(u(i, j - 1), u(i - 1, j)\big)\Big) + t(i, p[j]);$

$\}$

$\}$

$\#\boldsymbol{Valor\ del\ makespan}$

$z \leftarrow u[m - 1, n - 1]$

$\boldsymbol{return}\ (z)$

$\}$

3.3.2 swap

La funció swap té com a objectiu modificar una solució d'entrada intercanviant dos nodes d'aquesta. Aquesta funció ha estat desenvolupada en codi R.

El codi associat a la funció swap és el següent:

Variables d'entrada:

- Vector d'enters → Seqüència d'entrada v
- Valor enter → Node i
- Valor enter → Node j

Variables de sortida:

- Vector d'enters → Solució obtinguda v

$swap \leftarrow \boldsymbol{function}(v, i, j)\{$

$aux \leftarrow v[i]$

$v[i] \leftarrow v[j]$

$v[j] \leftarrow aux$

$\boldsymbol{return}\ (v);$

$\}$

3.3.3. TimeMatrixGenerator

La funció TimeMatrixGenerator té com a objectiu generar una matriu de temps a partir de dues matrius d'entrada, corresponents a les mitjanes i variàncies per a cada tasca i cada màquina de la instància d'entrada. Així doncs, aquesta funció simula uns temps d'entrada segons una distribució de probabilitat determinada per l'usuari. Aquesta és una de les funcions desenvolupades en C++ per a accelerar el rendiment del programa principal.

El codi associat a la funció TimeMatrixGenerator és el següent:

Variables d'entrada:

- Matriu d'enters → Matriu de mitjanes t
- Matriu d'enters → Matriu de variàncies var

Variables de sortida:

- Matriu d'enters → Matriu de temps $trandom$

$NumericMatrix\ \boldsymbol{TimeMatrixGenerator}(NumericMatrix\ t, NumericMatrix\ var)\{$

$\#\boldsymbol{Declaració\ de\ variables}:$

$int\ m = t.nrow(), n = t.ncol();$

$int\ i, j;$

$\#\boldsymbol{Inicialitzaci\acute{o}\ de\ la\ matriu\ de\ temps}$

$NumericMatrix\ trandom(m,n);$

$\#\boldsymbol{Generaci\acute{o}\ de\ matriu\ aleat\grave{o}ria}$

$\boldsymbol{for}(i=0;i<m;i++)\{$

$\quad \boldsymbol{for}(j=0;j<n;j++)\{$

$\quad\quad trandom(i,j) = (int)rnorm\big(1,t(i,j),var(i,j)\big)(0);$

$\quad \}$

$\}$

$\boldsymbol{return}(trandom);$

$\}$

3.3.4. VarMatrixGenerator

La funció VarMatrixGenerator té una funció similar a la funció desenvolupada a l'apartat 3.3.3, ja que la seva funció és generar una matriu de variàncies aleatòria, la qual es farà servir posteriorment per al càlcul de la matriu de temps. Així com els valors de les matrius de mitjanes que es fan servir han sigut extretes de les instàncies de Taillard (Veure apartat 4.1.1), hi ha poca informació disponible sobre valors de variàncies útils per al problema que es resol en aquest estudi. És per això que s'ha decidit fer servir una matriu de variàncies aleatòries emmarcades en un màxim i un mínim definit per l'usuari, basada en la bibliografia consultada. Aquesta funció ha estat desenvolupada en codi R, ja que no és una funció recurrent durant l'algoritme.

El codi associat a aquesta funció és:

Variables d'entrada:

- Nombre enter → Valor del límit inferior de variàncies $LowerBound$
- Nombre enter → Valor del límit superior de variàncies $UpperBound$
- Nombre enter → Nombre de màquines de la instància m
- Nombre enter → Nombre de tasques de la instància n

Variables de sortida:

- Matriu d'enters → Matriu de variàncies $VarMatrix$

$VarMatrixGenerator \leftarrow \boldsymbol{function}(LowerBound, UpperBound, m, n)\{$

$\#\boldsymbol{Generaci\acute{o}\ de\ la\ matriu\ de\ temps\ buida}$

$VarMatrix \leftarrow matrix(0, nrow = m, ncol = n)$

$\#\boldsymbol{Càlcul\ de\ la\ matriu\ de\ temps}$

$\boldsymbol{for}(a\ in\ 1{:}m)\{$

$\quad \boldsymbol{for}(b\ in\ 1{:}n)\{$

$\quad\quad VarMatrix[a,b] = runif(1, LowerBound, UpperBound)$

$\quad \}$

$\}$

$\boldsymbol{return}(VarMatrix)$

$\}$

3.3.5. EMS

La següent funció té com a objectiu realitzar una simulació on s'obtingui un valor mitjà de makespan, donades una matriu de temps, una matriu de variàncies i un valor d'iteracions determinat per l'usuari. La funció EMS ha estat desenvolupada en llenguatge R ja que no és una funció recurrent durant el programa principal.

El codi associat a aquesta funció és el següent:

Variables d'entrada:

- Matriu d'enters → Matriu de mitjanes per a cada tasca/màquina $MeanMAT$
- Matriu d'enters → Matriu de variàncies per a cada tasca/màquina $VarMAT$
- Vector d'enters → Seqüència solució inicial $InitialSeq$
- Nombre enter → Nombre d'iteracions per a la simulació $nREP$
- Variable binària → Variable per retornar els valors de makespan calculats $report.values$

Variables de sortida:

- Matriu d'enters → Matriu de variàncies $VarMatrix$

$EMS \leftarrow \boldsymbol{function}(MeanMAT, VarMAT, InitialSeq, nREP, report.values = FALSE)\{$

$\#\boldsymbol{Generació\ de\ la\ matriu\ on\ es\ guardaran\ els\ valors\ de\ makespan\ calculats}$

$MSValues \leftarrow matrix(0, nrow = 1, ncol = nREP)$

$\#\boldsymbol{Simulació\ del\ makespan}$

$\boldsymbol{for}(i\ in\ 1{:}nREP)\{$

$\quad \#\boldsymbol{Generació\ de\ matriu\ de\ temps\ aleatòria}$

$TIME \leftarrow TimeMatrixGeneratorC(MeanMAT, VarMAT)$

#Càlcul del makespan

$MS_{Calc} \leftarrow makespanC(TIME, InitialSeq)$

#Es guarda el valor de makespan al vector MSValues

$MSValues[i] = MS_{Calc}$

}

#Valor mitjà del makespan

$EMS_{Value} \leftarrow mean(MSValues)$

$\boldsymbol{if}(report.values)$

$\boldsymbol{return}\big(list(EMS_{Value} = EMS_{Value}, values = MSValues)\big)$

$\boldsymbol{else}$ {

$\boldsymbol{return}(EMS_{Value})$

}

Per a entendre el funcionament de la funció EMS, cal destacar alguns aspectes importants:

- El funcionament de la funció es basa en generar $nREP$ matrius de temps aleatòries amb els seus respectius valors associats de makespan. Aquests valors de makespan són guardats en un vector d'enters i finalment es calcula el valor mitjà dels valors continguts en aquest vector.
- Finalment, la funció pot retornar diversa informació, segons ho indiqui l'usuari. L'estat inicial de la variable binària $report.values$ és $FALSE$, de manera que només retornarà el valor mitjà de makespan EMS_{Value}. En cas contrari, si $report.values = TRUE$, la funció retornarà una llista amb el valor mitjà de makespan i tots els valors calculats durant el transcurs de la simulació.

3.3.6. NEHDet

La següent funció té com a objectiu aplicar la heurística NEH a una matriu de temps d'entrada determinada. La següent funció ha estat desenvolupada en llenguatge R ja que no és una funció recurrent en el programa principal.

El pseudocodi associat a aquesta heurística és:

Variables d'entrada:

- Matriu d'enters → Matriu de temps d'entrada $Instance$
- Funció objectiu f

Variables de sortida:

- Llista de variables → Llista amb la seqüència solució i el seu valor de makespan associat

$NEHDet \leftarrow \boldsymbol{function}(Instance)\{$

$\#\boldsymbol{Pas\ 1}$

$\#\boldsymbol{Generació\ de\ la\ seqüència\ inicial}$

$Seqüència\ Inicial \rightarrow Ordenar\ tasques\ en\ ordre\ decreixent\ de\ temps\ de\ procés$

$\#\boldsymbol{Pas\ 2}$

$\#\boldsymbol{Generació\ de\ la\ solució\ parcial\ inicial}$

$\boldsymbol{if}(makespanC[T_1 - T_2] \leq makespanC[T_2 - T_1])\ \{$

$sol \rightarrow Seqüència\ solució\ parcial\ inicial\ T_1 - T_2$

$\}$

$\boldsymbol{else}\ \{$

$sol \rightarrow Seqüència\ solució\ parcial\ inicial\ T_2 - T_1$

$\}$

$\#\boldsymbol{Pas\ 3-4}$

$\boldsymbol{for}(k\ in\ 3{:}n)\{$

$\boldsymbol{for}(i\ in\ 1{:}k)\{$

$test \leftarrow Càlcul\ de\ la\ seqüència\ solució\ parcial$

$testfit \leftarrow Càlcul\ del\ makespan\ parcial\ associat\ a\ test$

$\#\boldsymbol{Actualizar\ funció\ objectiu\ òptima}$

$\boldsymbol{if}(testfit < fit)\{$

$fit \leftarrow testfit$

$pos \leftarrow Posició\ del\ makespan\ òptim$

$\}$

$\}$

$sol \rightarrow Actualitzar\ solució\ associada\ al\ millor\ makespan\ parcial$

$\}$

$\#\boldsymbol{Calcular\ makespan\ associat\ a\ la\ solució\ final}$

$fit \leftarrow makespanC(Instance, sol)$

$\boldsymbol{return}$ $(Seqüència\ solució\ obtinguda\ (sol)\ i\ valor\ de\ makespan\ associat\ (fit))$

3.3.7. CDSHeuristic

La següent funció té com a objectiu desenvolupar la heurística CDS donada una matriu de mitjanes d'entrada i una matriu de variàncies d'entrada, la qual retorni una seqüència solució. Aquesta funció ha estat desenvolupada en llenguatge R ja que no és una funció recurrent en el programa principal.

El pseudocodi associat a la heurística CDS és el següent:

Variables d'entrada:

- Valor enter → Nombre de màquines m
- Valor enter → Nombre de tasques n
- Matriu d'enters → Matriu de mitjances per a cada màquina/tasca $MeanMat$
- Matriu d'enters → Matriu de variàncies per a cada màquina/tasca $VarMat$

Variables de sortida:

- Vector d'enters → Seqüència solució obtinguda mitjançant l'algoritme CDS $bestInitialSOL$

$CDSHeuristic \leftarrow \boldsymbol{function}(m, n, MeanMat, VarMat)\{$

$\#\boldsymbol{Inicialització\ de\ variables}$

$bestInitialMS \leftarrow 99999$

$actualInitialMS \leftarrow bestInitialMS$

$bestInitialSOL \leftarrow Generació\ de\ millor\ vector\ solució\ buit$

$\#\boldsymbol{Inici\ de\ la\ heurística\ CDS}$

$\boldsymbol{for}(k\ in\ 1{:}m-1)\{$

$\#\boldsymbol{Pas\ 1}$

$testMeanMat \leftarrow Càlcul\ de\ la\ matriu\ de\ temps\ virtuals$

$\#\boldsymbol{Pas\ 2}$

$seqSOL \leftarrow Calcular\ seqüència\ solució\ via\ regla\ de\ Johnson$

$\#\boldsymbol{Pas\ 3}$

$\#\boldsymbol{Actualitzar\ millor\ solució\ obtinguda}$

$\boldsymbol{if}(EMS[seqSOL] < EMS[bestInitialSOL])\ \{$

$bestInitialMS \leftarrow actualInitialMS$

$bestInitialSOL \leftarrow seqSOL$

$\}$

$\}$

$\boldsymbol{return}$ $(Seqüència\ solució\ obtinguda(bestInitialSOL))$

3.3.8. SADet

La següent funció té com a objectiu desenvolupar la metaheurística Simulated Annealing, donades una matriu de mitjanes i una solució inicial. Aquesta funció només necessita com a input una matriu de mitjanes, sense tenir en compte la variància, ja que explora únicament solucions deterministes per a que posteriorment es realitzi la simulació estocàstica pertinent. A diferència d'altres funcions recurrents que s'han descrit anteriorment, aquesta funció ha estat desenvolupada en llenguatge R tot i la seva recurrència durant el programa principal, ja que les funcions a les quals recorre la funció SADet ja estan desenvolupades en llenguatge C++, de manera que s'ha considerat que la millora en el rendiment de la funció no seria substancial.

El pseudocodi associat a l'algoritme SA és el següent[15]:

Variables d'entrada:

- Matriu d'enters → Matriu de mitjanes per a cada màquina/tasca mat
- Vector d'enters → Solució inicial $sini$
- Valor enter → Nombre d'iteracions T_{max} (Condició de STOP)
- Valor enter → Valor del paràmetre μ (mu)
- Variable binària → El valor $report.evol$ permet definir a l'usuari si vol que el programa retorni els valors de makespan obtinguts durant l'evolució de l'algoritme

Variables de sortida:

- Llista de variables → Llista amb la seqüència solució obtinguda sol, el valor de makespan associat a la solució obj i un vector d'enters $evol$ que conté els valors calculats de makespan durant l'algoritme.

$SADet \leftarrow \boldsymbol{function}(mat, sini, T_{max}, mu, report.evol = TRUE)\{$

$sol \leftarrow sini$

$obj \leftarrow Valor\ de\ makespan\ de\ la\ solució\ inicial, calculat\ via\ la\ funció\ makespanC$

$solbest \leftarrow sol$

$objbest \leftarrow obj$

$T \leftarrow T_{max}$

while $T > 0$ {

$soltest \leftarrow Selecciona\ una\ solució\ veïna\ mitjançant\ la\ funció\ swap$

$objtest \leftarrow Makespan\ de\ la\ seqüència\ soltest\ calculat\ via\ la\ funció\ makespanC$

if $(obj < objtest)$ {

$solbest \leftarrow sol$

$objbest \leftarrow obj$

}

else {

#***Si es compleix el criteri d'acceptació***

s'acceptarà la solució candidata

if $\left(e^{-\frac{\mu(objtest-obj)}{T}} > U(0,1)\right)$ {

$sol \leftarrow soltest$

$obj \leftarrow objtest$

}

}

#***Guardar valor de makespan actual al vector evol***

$evol[T_{max} - T] \leftarrow obj$

$T \leftarrow T - 1$

}

if $(report.evol)$ {

return $\begin{pmatrix} Retorna\ seqüència\ solució\ (sol), valor\ de\ makespan\ associat(obj) \\ i\ vector\ d'evolució\ de\ makespan\ (evol) \end{pmatrix}$

}

else {

return $(Retorna\ seqüència\ solució\ (sol)\ i\ valor\ de\ makespan\ associat\ (obj))$

}

3.3.9. TSDet

La següent funció té com a objectiu desenvolupar l'algoritme Tabu Search, donades una matriu de mitjanes d'entrada i una solució inicial. De la mateixa manera que passa amb la funció de SA, aquesta funció explora solucions basades en valors de la funció objectiu deterministes. D'altra banda, també ha estat desenvolupada mitjançant el llenguatge de programació R, ja que les funcions recurrents que s'executen ja estan desenvolupades en llenguatge C++.

El pseudocodi d'aquesta funció és el següent[15]:

Variables d'entrada:

- Matriu d'enters → Matriu de mitjanes per a cada màquina/tasca mat
- Vector d'enters → Solució inicial $sini$
- Valor enter → Nombre d'iteracions $iter$ (Condició de STOP)
- Valor enter → Valor del paràmetre $tabu.size$
- Variable binària → El valor $report.evol$ permet definir a l'usuari si vol que el programa retorni els valors de makespan obtinguts durant l'evolució de l'algoritme

Variables de sortida:

- Llista de variables → Llista amb la seqüència solució obtinguda sol, el valor de makespan associat a la solució obj i un vector d'enters $evol$ que conté els valors calculats de makespan durant l'algoritme.

$TSDet \leftarrow \boldsymbol{function}(mat, sini, iter, tabu.size, report.evol = TRUE)\{$

$sol \leftarrow sini$

$obj \leftarrow Valor\ de\ makespan\ de\ la\ solució\ inicial, calculat\ via\ la\ funció\ makespanC$

$solbest \leftarrow sol$

$objbest \leftarrow obj$

$k \leftarrow -1$

$tabu.list \leftarrow Inicialitzar\ llista\ tabú$

$\boldsymbol{while}\ k \leq iter\ \{$

$soliter \leftarrow Trobar\ seqüència\ solució\ d'entre\ el\ conjunt\ de\ solucions$

$veïnes\ N(s)\ que\ tingui\ un\ valor\ de\ makespan\ associat\ mínim$

$\#\boldsymbol{Actualitzar\ solució\ a\ explorar}$

$sol \leftarrow soliter$

$obj \leftarrow objiter$

$\#\boldsymbol{Actualitzar\ millor\ soluci\acute{o}\ trobada}$

$\boldsymbol{if}(obj < objbest)\{$

$objbest \leftarrow obj$

$solbest \leftarrow sol$

$\}$

$\#\boldsymbol{Actualitzar\ llista\ tab\acute{u}}$

$\boldsymbol{if}(Condició\ actualitzar\ llista\ tabú = TRUE)\{$

$tabu.list \leftarrow Actualitzar\ llista\ tabú$

$\}$

$\#\boldsymbol{Actualitzar\ vector\ d'evolució\ del\ makespan}$

$evol[k] \leftarrow obj$

$k \leftarrow k + 1$

$\}$

$\boldsymbol{if}(report.evol)\{$

$\boldsymbol{return}\ \begin{pmatrix} Retorna\ seqüència\ solució\ (sol), valor\ de\ makespan\ associat(obj) \\ i\ vector\ d'evolució\ de\ makespan\ (evol) \end{pmatrix}$

$\}$

$\boldsymbol{else}\{$

$\boldsymbol{return}\ (Retorna\ seqüència\ solució\ (sol)\ i\ valor\ de\ makespan\ associat\ (obj))$

$\}$

3.3.10. Perturbation

La següent funció té com a objectiu pertorbar la solució actual quan la metaheurística SIMILS ho requereixi. Donada la matriu de temps d'entrada i la seqüència a pertorbar, es retorna una seqüència que es convertirà en solució candidata. Aquesta funció ha estat desenvolupada en llenguatge de programació R ja que, tot i ser una funció força recurrent, no s'ha considerat que la seva recurrència sigui suficient com per afectar negativament al rendiment del programa principal.

El pseudocodi d'aquesta funció és el següent:

Variables d'entrada:

- Matriu d'enters → Matriu de mitjanes per a cada màquina/tasca $MeanMat$
- Vector d'enters → Solució inicial $inputsol$

Variables de sortida:

- Llista de variables → Llista amb la seqüència solució obtinguda $bestsol$ i el valor de makespan associat a la solució obj.

$Perturbation \leftarrow \boldsymbol{function}(inputsol, MeanMat)\{$

$\#\boldsymbol{Inicialitzaci\acute{o}\ de\ variables}$

$actualsol \leftarrow inputsol$

$bestsol \leftarrow actualsol$

$actualMS \leftarrow Càlcul\ del\ makespan\ de\ la\ solució\ inicial\ via\ la\ funció\ makespanC$

$bestMS \leftarrow actualMS$

$move \leftarrow Seleccionar\ 2\ nodes\ aleatoris\ de\ la\ solució\ inicial\ inputsol$

$\#\boldsymbol{Intercanvi\ \ de\ posicions\ millorat}$

$\boldsymbol{for}\ (i\ in\ 1{:}\ 2)\ \{$

$\#\boldsymbol{Intercanviar\ posicions}$

$actualsol \leftarrow Actualitzar\ solució\ intercanviant$

$els\ 2\ nodes\ seleccionats\ (vector\ move)$

$\#\boldsymbol{Desplaçament\ de\ nodes\ a\ l'esquerra}\ (\boldsymbol{Operador\ shift\ to\ left})$

$bestsol \leftarrow S'explora\ la\ millor\ solució\ possible\ aplicant\ l'operador\ shift\ to\ left$

$\}$

$\boldsymbol{return}\ (Retorna\ seqüència\ solució\ (bestsol)\ i\ valor\ de\ makespan\ associat\ (obj))$

$\}$

3.3.11. SIMILS

A continuació es desenvolupa l'algoritme de simulació combinat amb cerca local iterada (ILS). Aquesta funció engloba la resta de funcions descrites anteriorment i serà la funció que es farà servir per resoldre el problema de flux de tasques amb temps estocàstics mitjançant diferents heurístiques. És doncs a partir d'aquí, des d'on es basa l'experiment computacional objectiu d'aquest estudi. El codi d'aquesta funció ha estat desenvolupat en llenguatge R ja que les

funcions recursives que en formen part ja estan desenvolupades en llenguatge C++, per tal d'accelerar el seu rendiment.

El pseudocodi de la funció és el següent:

Variables d'entrada:

- Matriu d'enters → Matriu de mitjanes per a cada màquina/tasca $MeanMat$
- Matriu d'enters → Matriu de variàncies per a cada màquina/tasca $VarMat$
- Valor enter → Nombre d'iteracions per a la simulació curta $nREP$
- Valor enter → Nombre d'iteracions per a la simulació llarga $LONGREP$
- Valor enter → Nombre d'iteracions de l'algoritme ILS $TREPBase$
- Valor enter → Selector de solució inicial $InitialSolSelector$
- Valor enter → Selector d'algoritme de cerca local $LocalSearchSelector$
- Valor enter → Valor del paràmetre μ (mu)
- Valor enter → Nombre d'iteracions per a l'algoritme de cerca local $IterMax$
- Valor enter → Valor de la dimensió de la llista tabú $tabusize$
- Valor enter → Nombre de màquines m
- Valor enter → Nombre de tasques n
- Vector d'enters → Solució inicial $inputsol$

Variables de sortida:

- Llista de variables → Llista amb les dades necessàries per a l'anàlisi de la solució global del problema: $EMS_{Value}, values, MinMSStoch, MaxMSStoch, VarianciaMS, CampanaMS\ i\ sol.$

$SIMILS \leftarrow \boldsymbol{function}(MeanMat, VarMat, nREP, LONGREP, TREPBase,$

$InitialSolSelector, LocalSearchSelector, muValue, IterMax, tabusize, m, n)\{$

#***Inicialització de les solucions***

$solucions \leftarrow Es\ crea\ una\ llista\ que\ contingui\ el\ valor\ mitjà\ de\ makespan\ obtingut\ en\ la\ simulació\ llarga\ (EMS_{Value}), els\ valors\ de\ makespan\ obtinguts\ durant\ la\ simulació\ (values), el\ valor\ mínim\ de\ makespan\ obtingut\ en\ la\ simulació\ (MinMSStoch), el\ valor\ màxim\ de\ makespan\ obtingut\ en\ la\ simulació\ (MaxMSStoch), el\ valor\ de\ la\ variància\ resultant\ dels\ valors\ de\ makespan\ calculats\ durant\ la\ simulació\ (VarianciaMS), els\ valors\ associats\ de\ probabilitat\ en\ una\ distribució\ de\ probabilitat\ normal\ (CampanaMS)\ i\ la\ seqüència\ solució\ obtinguda\ durant\ el\ desenvolupament\ de\ l'algoritme\ (sol).$

#***Càlcul de la solució inicial***

if $(InitialSolSelector == 1)\{$

$baseSOL \leftarrow Calcular\ la\ solució\ inicial\ mitjançant\ la\ heurística\ NEH$

$\}$

if $(InitialSolSelector == 2)\{$

$baseSOL \leftarrow Calcular\ la\ solució\ inicial\ mitjançant\ la\ heurística\ CDS$

$\}$

#***Cerca local***

if $(LocalSearchSelector == 1)\{$

$baseSOL \leftarrow Aplicar\ SA\ a\ baseSOL$

$lastImprovement \leftarrow Valor\ de\ makespan\ resultant\ de\ SA$

$\}$

if $(LocalSearchSelector == 2)\{$

$baseSOL \leftarrow Aplicar\ TS\ a\ baseSOL$

$lastImprovement \leftarrow Valor\ de\ makespan\ resultant\ de\ TS$

$\}$

#***Actualitzar millor solució determinista***

$bestDetSOL \leftarrow baseSOL$

#***Inicialitzar flag d'actualització de solució i nombre d'iteracions***

$ACflag \leftarrow 1$

$TREP \leftarrow TREPBase$

while $(TREP > 0)\ \{$

#***Pertorbar solució actual***

$newSOL \leftarrow Obtenir\ nova\ solució\ aplicant\ la$

$funció\ Perturbation\ a\ la\ solució\ actual$

#***Cerca local***

if $(LocalSearchSelector == 1)\{$

$newSOL \leftarrow Aplicar\ SA\ a\ newSOL$

$\}$

$$\boldsymbol{if}(LocalSearchSelector == 2)\{$$

$$newSOL \leftarrow Aplicar\ TS\ a\ newSOL$$

$$\}$$

$$\#\boldsymbol{Calcular\ makespan}$$

$$newSOLMS \leftarrow Calcular\ makespan\ per\ a\ newSOL$$

$$baseSOLMS \leftarrow Calcular\ makespan\ per\ a\ baseSOL$$

$$\#\boldsymbol{Actualitzar\ soluci\acute{o}\ actual}$$

$$\boldsymbol{if}(newSOLMS < baseSOLMS)\{$$

$$ACflag \leftarrow 0$$

$$lastimprovement \leftarrow baseSOLMS$$

$$baseSOL \leftarrow newSOL$$

$$bestDetSOLMS \leftarrow Calcular\ makespan\ per\ a\ bestDetSOL$$

$$\#\boldsymbol{Actualitzar\ millor\ soluci\acute{o}\ determinista}$$

$$\boldsymbol{if}(newSOLMS < baseSOLMS)\{$$

$$bestDetSOL \leftarrow newSOL$$

$$testStochMS \leftarrow Fer\ simulació\ curta\ de\ makespan\ per\ a\ bestDetSOL$$

$$\#\boldsymbol{Actualitzar\ millor\ soluci\acute{o}\ estoc\grave{a}stica}$$

$$\boldsymbol{if}(testStochMS < bestStochSOLMS)\{$$

$$bestStochSOL \leftarrow bestDetSOL$$

$$bestStochSOLMS \leftarrow testStochMS$$

$$\}$$

$$\}$$

$$\}\ \boldsymbol{else}\ \{$$

$$\#\boldsymbol{Criteri\ d'acceptaci\acute{o}}$$

$$\boldsymbol{if}(AClag == 0\ \&baseSOLMS < lastimprovement)\{$$

$$baseSOL \leftarrow newSOL$$

$$lastimprovement \leftarrow baseSOLMS$$

$$ACflag \leftarrow 1$$

$$\}$$

$$\}$$

$$TREP \leftarrow TREP - 1$$

$$\}$$

#Simulació de resultats

$$ComputeStochasticSolution \leftarrow Realitzar\ simulació\ llarga\ de\ makespan\ per\ a\ la\ millor\ solució\ estocàstica\ obtinguda\ en\ l'algoritme\ bestStochSOL$$

$$solucions \leftarrow Guardar\ variables\ d'interès\ resultants\ de\ la\ simulació\ llarga$$

$$\boldsymbol{return}\ (solucions)$$

$$\}$$

4. Resolució del problema mitjançant heurístiques

L'últim pas per a completar l'estudi és estructurar el programa principal en R que reculli el codi desenvolupat fins al moment, d'una manera lògica i ordenada, per tal d'arribar a assolir l'objectiu principal d'aquest estudi, que és resoldre el problema de flux de tasques amb temps estocàstics a través de diferents heurístiques. La estructura del programa principal es pot visualitzar a la Figura 18. A través dels propers apartats, es definirà cada un dels punts que conformen aquesta estructura.

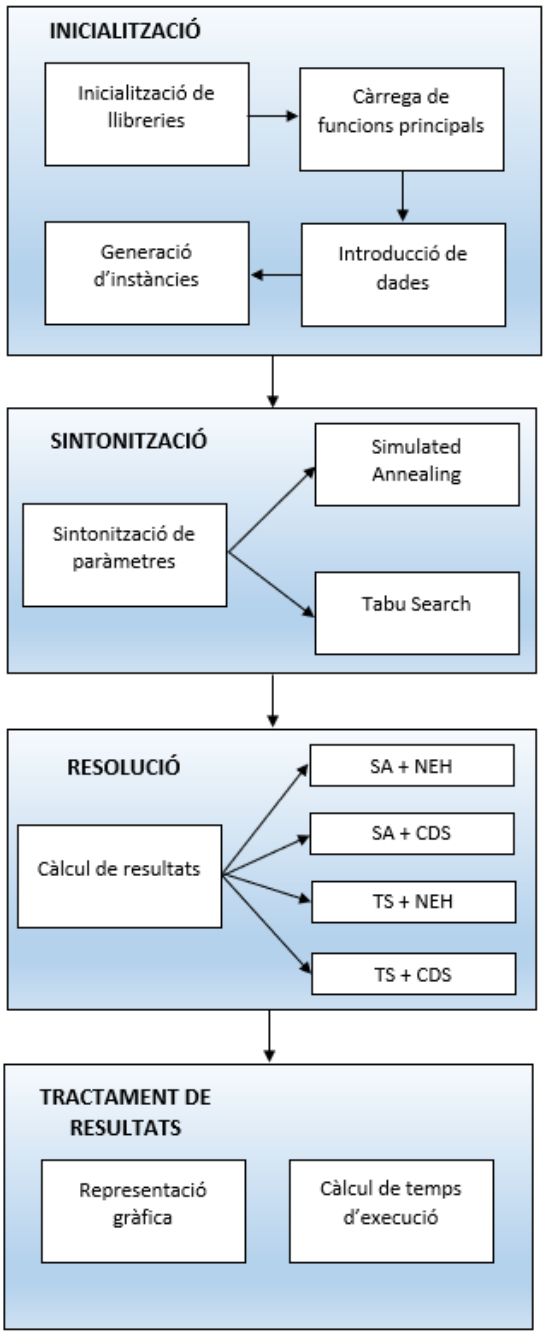

Figura 18. Estructura del programa principal

4.1. Inicialització

Prèviament a l'execució de les diferents heurístiques, és necessari realitzar una sèrie de passos previs per al seu correcte funcionament. El primer pas és carregar les llibreries necessàries:

- **Llibreria "dplyr":** Aquest paquet de codi bàsic permet manipular de manera fàcil i eficient estructures de dades dins del codi. Proveeix l'usuari de funcions predefinides com per exemple select(), summarise() o sample().
- **Llibreria "ggplot2":** Aquest paquet de codi permet representar una gran varietat de gràfics diferents, que resulten interessants per a representar els resultats de l'estudi.
- **Llibreria "Rcpp":** Aquest paquet de codi, que ja ha estat definit a l'apartar anterior, permet "traduir" línies i funcions de codi de C++ a R, sense afectar a la funcionalitat del programa principal.

Un cop carregades les llibreries necessàries, es carreguen al workspace les funcions desenvolupades. Aquestes funcions han estat desenvolupades en un R Script a part del programa principal per facilitar la comprensió del programa principal i seran executades a mesura que el programa ho sol·liciti. Per a carregar les funcions, es fa servir la ordre source("Functions.R").

El següent pas és introduir les dades necessàries per a la resolució del problema, les quals han de poder ser identificades de manera ràpida per tal de poder modificar-les fàcilment en cas que es vulgui. Aquests paràmetres no són necessàriament fixes, a diferència dels paràmetres μ, tabusize i nombre d'iteracions dels algoritmes de cerca local, que són fixats després de la sintonització d'aquests algoritmes. Els paràmetres que entren dins d'aquest grup de variables que poden ser modificades per l'usuari són:

- $LowerBound$: Límit inferior de la variància per a totes les màquines i tasques.
- $UpperBound$: Límit superior de la variància per a totes les màquines i tasques.
- $nREP$: Nombre d'iteracions per a una simulació curta de EMS.
- $LONGREP$: Nombre d'iteracions per a una simulació llarga de EMS.
- $TREPBASE$: Nombre d'iteracions per a l'algoritme SIMILS.

4.1.1. Generació d'instàncies

Finalment, es procedeix a generar les instàncies que es resoldran durant l'estudi. Cada instància ve definida per un nombre de màquines m, un nombre de tasques n i els valors de temps per a cada màquina i tasca t_{ij}. Cada màquina i tasca té associat un valor de temps t_{ij}, els quals estan definits per una distribució de probabilitat normal amb mitjana μ_{ij} i variància σ_{ij}^2. A la Figura 19 es pot observar gràficament la distribució de probabilitat que segueixen els temps t_{ij}.

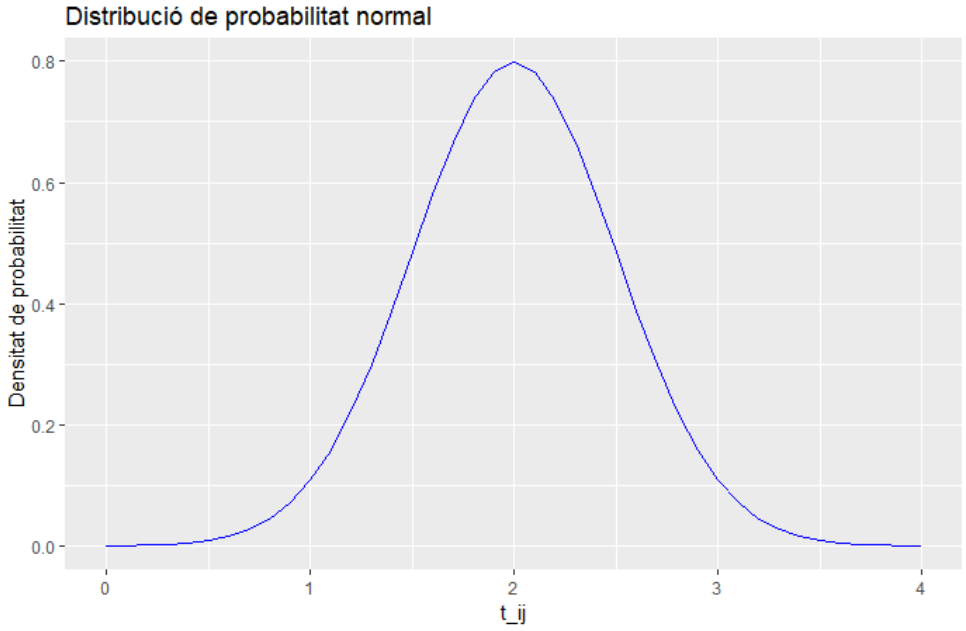

Figura 19. Densitat de probabilitat d'una distribució normal amb μ = 2 i σ² = 0,5

Els valors mitjans μ_{ij} i les variàncies σ_{ij}^2 per a cada parella màquina/tasca estan recollits en una matriu de mitjanes $MeanMat$ i una matriu de variàncies $VarMat$, respectivament. Formalment, ambdues matrius es defineixen de la següent manera:

$$MeanMat = \begin{pmatrix} \mu_{11} & \dots & \mu_{1n} \\ \dots & \dots & \dots \\ \mu_{m1} & \dots & \mu_{mn} \end{pmatrix}$$

$$VarMat = \begin{pmatrix} \sigma_{11} & \dots & \sigma_{1n} \\ \dots & \dots & \dots \\ \sigma_{m1} & \dots & \sigma_{mn} \end{pmatrix}$$

Per una banda, les matrius de mitjanes $MeanMat$ per a cada dimensió n i m han estat obtingudes de manera directa del benchmark desenvolupat per E.Taillard[19], el qual està sintetitzat en l'arxiu R "TaillardFS.RData". Cal destacar que les instàncies de Taillard són un conjunt de 120 instàncies agrupades en funció de la dimensió de la instància. Per a cada dimensió de la instància s'inclouen 10 mostres diferents, cada una d'elles representada amb una matriu de temps t_{ij} diferents i, en conseqüència, cada una d'elles té una seqüència solució diferent. A la Taula 19 es resumeixen el nombre de mostres de Taillard per a les diferents dimensions de les instàncies.

	n = 20	n = 50	n = 100	n = 200	n = 500
m = 5	10	10	10	-	-
m = 10	10	10	10	10	-
m = 20	10	10	10	10	10

Taula 19. Nombre de mostres de Taillard per a cada valor m i n

En el present estudi s'ha fet servir 1 mostra de temps t_{ij} per a cada dimensió d'instància amb $m = 5, 10\ i\ 20$ i $n = 20, 50\ i\ 100$, resultant un total de 9 instàncies diferents per analitzar el rendiment de les diferents heurístiques. Les instàncies amb dimensió $n = 200$ i $n = 500$ han estat descartades per al present estudi a causa de les seves dimensions, ja que la resolució d'aquest tipus d'instàncies suposa un consum de temps de computació massa elevat amb els recursos computacionals disponibles per a realitzar l'estudi. Els valors numèrics de les 9 instàncies de Taillard que s'usen en l'anàlisi són els següents:

```
$tij
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
[1,]   54   83   15   71   77   36   53   38   27    87    76    91    14    29    12    77    32    87    68    94
[2,]   79    3   11   99   56   70   99   60    5    56     3    61    73    75    47    14    21    86     5    77
[3,]   16   89   49   15   89   45   60   23   57    64     7     1    63    41    63    47    26    75    77    40
[4,]   66   58   31   68   78   91   13   59   49    85    85     9    39    41    56    40    54    77    51    31
[5,]   58   56   20   85   53   35   53   41   69    13    86    72     8    49    47    87    58    18    68    28
```

Figura 20. MeanMat205 (m = 5 i n = 20)

```
$tij
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
 [1,]   74   21   58    4   21   28   58   83   31    61    94    44    97    94    66     6    37    22    99    83
 [2,]   28    3   27   61   34   76   64   87   54    98    76    41    70    43    42    79    88    15    49    72
 [3,]   89   52   56   13    7   32   32   98   46    60    23    87     7    36    26    85     7    34    36    48
 [4,]   60   88   26   58   76   98   29   47   79    26    19    48    95    78    77    90    24    10    85    55
 [5,]   54   66   12   57   70   82   99   84   16    41    23    11    68    58    30     5     5    39    58    31
 [6,]   92   11   54   97   57   53   65   77   51    36    53    19    54    86    40    56    79    74    24     3
 [7,]    9    8   88   72   27   22   50    2   49    82    93    96    43    13    60    11    37    91    84    67
 [8,]    4   18   25   28   95   51   84   18    6    90    69    61    57     5    75     4    38    28     4    80
 [9,]   25   15   91   49   56   10   62   70   76    99    58    83    84    64    74    14    18    48    96    86
[10,]   15   84    8   30   95   79    9   91   76    26    42    66    70    91    67     3    98     4    71    62
```

Figura 21. MeanMat2010 (m = 10 i n = 20)

```
$tij
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
 [1,]   50   90   39   34   66   81   27   48   46    68    48    92    78    84    93    39    43     1    65    87
 [2,]   78   56    9   43   84   73   66   38   83    57    97    52    77    13    12     2    65    93    39     1
 [3,]   36   43   10   19   55   48   85   70   82    39    91    82    85    17     6    54    87    85     4    72
 [4,]   85   88   60   98    4   99   53   21   33    53    63    18    45    29    43    41    80     4    31    19
 [5,]    9   92   98   44   51    8   31   15   47    31    80    83    20    84    69    49    93    39    13    88
 [6,]   75   64   96   95   22   41   26   33   68     9    81    28    61    69    37    57    36    80    96    74
 [7,]   46   94    6   19   20   51   85   92   43    75    70    70    36    31    76    63    89    46    25    88
 [8,]   73    3   56   73   80   82   36   98   90    46    10    46    65    83    75    47    61    28    59    22
 [9,]   71   49   36   87    8   25   76   73   80     6     6    33    79    10    93    65    26    73    42    18
[10,]    7   40   33   64    5   25   89   95   58    83    28    35    74     5     6     9     3     2    35    41
[11,]   49   49   15   18   65   55    1   79   10    37    77    80    79    84    93    21    85    64    46    35
[12,]    3   53   59    7   65   58   24   55   26    40    89    94    51    74    54    86    22    83    19    44
[13,]   60   88   15   26   11   16   55   59   81    53    92    23    55    79    13    89     2    17    97    41
[14,]   12   47   46   17   43   16   91   94   73    89    12    58    25    24    55     1    67     3     1    71
[15,]   75   19   60   87   27   48   72   88   48    59    74    86    49    94    15    95    41    94    15    71
[16,]   31   61   47   32   34   69   32    1    1    80    19    57    98    37    31    51    66    38    62    72
[17,]   70   78   41    9   47   94   26   65   17    42    59    80     7    75    63    96     7    10    47    38
[18,]   20   78   38   26   64   62   11   38   68    37    74     9    65    16    38    85    50    62    39    97
[19,]   88   30   34   33   21    7   94   10   73    85    82    62    99    67    61    10     4    70    31    49
[20,]    9   41   22   34   83   55    3    8   75    30    57    65    89    60    90    84    74    17     2    19
```

Figura 22. MeanMat2020 (m = 20 i n = 20)

```
$tij
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
[1,]   75   87   13   11   41   43   93   69   80    13    24    72    38    81    83    88    26     6    89    67    70    30    89    30
[2,]   26   37   25   95   49   12   59   17   46    20    52    44    92    75    95    33    10    45     2    62    62    82    29    29
[3,]   48    4   92   92   72   45    5   98   93    17    79    11    16    89    81    92    45    61    39    28    94    87    23     1
[4,]   26   67    4   14   93   54   21   20    6    18    75    25    16    77    28    24    15    77    36    16    32    46    21    81
[5,]   77   94    9   57   29   79   55   73   65    86    25    39    76    24    38     5    91    29    22    27    39    31    46    18
     [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
[1,]   68   21   78   46   99   10   17   23   83   47   86   18   67   46    4   14    4   20   88   50   84   58
[2,]   94   20   42   80   94   35    8   41   65    4   71   30   14   32   50   30   27   98   39   84   65   12
[3,]   55   91   67   91    4   60   38   25   90   93   13   65   25   34   47   98   91   11   46   50   77    5
[4,]   28   70   89   54   96   62   46   60   19   97   13    7   44    7   73   15   66   70   97   33   97   64
[5,]   93   58   85   58   97   10   79   93    2   87   17   18   10   50    8   26   14   21   15   10   85   46
     [,47] [,48] [,49] [,50]
[1,]   93   76   50   30
[2,]   58   45   49   15
[3,]   14   47   80   45
[4,]   73   28    4   87
[5,]   42   18   36    2
```

Figura 23. MeanMat505 (m = 5 i n = 50)

```
$tij
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
 [1,]   46   52   79   45   97   10   44   24   85    75    66    49    95    61    19    47    84    13    11    19    98     2    85    44
 [2,]   61   87   51   25   73   93   28   90   94    59    64     2    16    35    53    40    81    26    85     4     4    10    63    96
 [3,]    3    1   58   85   33   71   58   56   64    43    48    69    96    35    82    53    64    11    61    36    53    87    88    10
 [4,]   51   24   21   57   69   51   50   51   21    19    63    91    11     6    31    63    36    39    57    47    56    65    59     4
 [5,]   37   16   42   47   94   14   94   34   72    36    88    51    41    71    94    99    11    97    44    77    69    91    38    25
 [6,]   79   93   68   75   37   44   34   39   76    62    74    28    78    43    98    83    91    27     6    82    60    44    43    76
 [7,]   83   87   38   38   86   67   23   19   97    78    66    67     7    23    67     8    77    71    85    29    49     3    94    76
 [8,]   22   29   99   25   98   55   80   82   33    68    47    74    26    61    95    55    11    42    72    14     8    98    90    36
 [9,]   27   92   75   94   18   41   37   58   56    20     2    39    91    81    33    14    88    22    36    65    79    23    66     5
[10,]   24   47   39   66   41   46   24   23   68    50    93    22    64    81    94    97    54    82    11    91    23    32    26    22
      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
 [1,]     7    73    19    69    12    73    85    23    53    16    88     8    26    42    58    63     7     2    44    38    24    76
 [2,]    55    71    66    94     7    15    11    99    37    50    56    69    22    56    67    63    96    74     4    42    40    30
 [3,]    32    38    25    24    90     7    11    49     2    76    17    32    39     9    83    69    67    28    88    23    91    71
 [4,]    10    12    62    43    49    54    87    29     2    18    75    39    77    69    15    78    68    37    22    41    92    67
 [5,]    87     7    66    54    86    49     3    48    44    93    37    82    31    59    78    33    36     3    58    10    98     6
 [6,]    99    66    11    35    52     8    40    62    25    24    30     1    73    27    16    91    33    11    99     2    60    90
 [7,]    95    48     4    37    82    57    61     6    97     5    27    95    46    92    46    52     8    11     7    54    72    57
 [8,]    75    69    26    24    55    98    86    30    92    94    66    47     3    41    41    47    89    28    39    80    47    57
 [9,]    15    51     2    81    12    40    59    32    16    87    78    41    43    94     1    93    22    93    62    53    30    34
[10,]    12    23    34    87    59     2    38    84    62    10    11    93    57    81    10    40    62    49    90    34    11    81
      [,47] [,48] [,49] [,50]
 [1,]    85    61    32    90
 [2,]    93    36    25    87
 [3,]     3    26    41    96
 [4,]    24    87    91    31
 [5,]    44    62    24    94
 [6,]    36    62    15     3
 [7,]    85    22    87    65
 [8,]    74    38    59     5
 [9,]    27    30    54    77
[10,]    51    21    39    27
```

Figura 24. MeanMat5010 (m =10 i n = 50)

```
$tij
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
 [1,]   52   95   42   75   44   57   89   53   84    62    91    14    95    89     4    95     2    97    68    20    33    51    98     8
 [2,]   63   99   69   70   53   21   10   31   80    18     5    18    17    71    90    93    14    49    52     7    78    57    41    75
 [3,]   82   21   79   95   46   23   40   95   87    37    24    24    65    62    19    67    66     6    65    59     2    67    82    90
 [4,]   16   26   46   66   76   31   36    8   37    21     3    76    67     5    47    72    66    56    95    49    47    26    81    56
 [5,]   63   55   59   35   21   59   78   25   30    38    78    79    58    44    38    76    70    72    85     8    10    84    42    67
 [6,]   94   34   89   62   47   66   76   15   18    54    24    55    96    10    12    96    53    92    77     6    91    14    41    30
 [7,]   79   21   93   32    8   45   37   78   26    98    17    25    21    28    68    24    62    89    60    64    38    90    87     1
 [8,]   22    6   24   55   48   57   78    5   50    83    70    21    71    58    36    50    31    86    29    30    93    49    83    89
 [9,]   80   13   64   77   17   78   82    4   72    93    68    25    67    80    43    93    21    33    14    30    59    83    85    85
[10,]   96    3   50   57   66   84   98   55   70    32    31    64    11     9    32    58    98    95    25     4    45    60    87    31
[11,]   53   19   99   62   88   93   34   72   42    65    39    79     9    26    72    29    36    48    57    95    93    79    88    77
[12,]   54   67   25   77   38   98   96   20   15    36    65    97    27    25    61    24    97    61    75    92    73    21    29     3
[13,]   71   90   59   82   22   88   35   49   78    69    76     2    14     3    22    26    44     1     4    16    55    43    87    35
[14,]   27   93   49   63   65   34   10   56   51    97    52    46    16    50    96    85    61    76    30    90    42    88    37    43
[15,]   95   53   54   22   84   54    2   80   84    66    25    16    79    90    51    29    29    90    83    83    19    95    87    12
[16,]    3   80   78   32   53   43   85   19   48    49    66    22    37    51    82    59    88    77    19    32    52     9    96    23
[17,]   92   62   11   83   87   66   98   42   23    45    52     6     3    64    55    97    83    42    81    92    68    46    56    88
[18,]   80   38   55   34   85   44   47   66   19    66    61    60    98    82    79    71    28    74    27    33    13     9    12    51
[19,]   61   86   16   42   14   92   67   77   46    41    78     3    72    95    53    59    34    66    42    63    27    92     8    65
[20,]   74   38    4   31   62   39   97   57    9    54    13    47     6    70    19    97    41     1    57    60    62    14    90    76
      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
 [1,]   85   86   73    4   40   98   12   59   44   46    2   41   28   83   28   21   80   71    4   60   34   55
 [2,]   98   93   33   75   68   33   60   82   24   99    4   97   24   50   55   91   46   58   17   47   82    6
 [3,]   30   63    5   93   53   85   81   73   34   74   13   78   35   20   16   48   12   11   80    9   24   76
 [4,]   76   66   36   53   26   52   29   36   68   21   71   61   71   69   28   86   27   41   86   55   17   62
 [5,]   20   24   75   23   33   60   20   75   83   26   92   29   39   14   74   66   86   10   27    8    7   97
 [6,]   85   17   23   60   76   39   85   10   65   15   55   41   28   93   88   27   77   81   19   76   55   67
 [7,]   99   34    9   22   74   14   14   84   75   37   32   29   32   89   12   47   19   97    7   12   43   89
 [8,]   44   38   62   45   22   85   39   98   56   68   84   77   67   53   46   24   52   96    2   88   33   27
 [9,]   70   35    2   76   46   72   69   46    3   57   71   77   33   49   59   82   59   70   76   10   65   19
[10,]    1   96   22   95   73   77   30   88   14   22   93   48   10    7   14   91    5   43   30   79   39   34
[11,]   94   39   74   46   17   30   62   77   43   98   48   14   45   25   98   30   90   92   35   13   75   55
[12,]   96   51   26   44   56   31   64   38   44   46   66   31   48   27   82   51   90   63   85   36   69   67
[13,]   76   98   78   81   48   25   81   27   84   59   98   14   32   95   30   13   68   19   57   65   13   63
[14,]   88   91   14   63   65   74   71    8   39   95   82   17   38   69   17   24   66   75   52   59    4   73
[15,]   34   23   44   30   82   83   42   56   89   38   96   10    3   53   97   11   65   47   76   22   17   14
[16,]   64   22   37    3   52   44   11   21   85    6   40   68   30   35   58   31   11   11    6   59   64   65
[17,]   50   13   23   13   49   18   50   94   71   64   31   21    2   63   58   36   64   52    8   94   51   36
[18,]   16   49   83   48   13   78   96   77   68   88   77   76   73   92   72   87   66   98   40   31   75   45
[19,]   34    6   42   39    2    7   85   32   14   74   59   95   48   37   59    4   42   93   32   30   16   95
[20,]   12   89   37   35   91   69   55   48   56   84   22   51   43   50   62   61   10   87   99   40   91   64
      [,47] [,48] [,49] [,50]
 [1,]   53   96   37   37
 [2,]   15   91   74   42
 [3,]   32   35   66   48
 [4,]   96   59   53   93
 [5,]   84   56   61    9
 [6,]   65    8   18   56
 [7,]   14   33   56   57
 [8,]   49   78   82   65
 [9,]   77   86   21   75
[10,]   77   81   11   10
[11,]   80   67    3   93
[12,]   81   18   81   72
[13,]   26   96   53   94
[14,]   56   19   39   51
[15,]   11   69   91   53
[16,]   23   80   75   63
[17,]   82   30   17   21
[18,]   98   90    4   23
[19,]   58   12   95   21
[20,]   62   53   33   16
```

Figura 25. MeanMat5020 (m = 20 i n = 50)

```
$tij
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
[1,]   73   84   57   52   66   67   33   62   65     7     6    31    42    82    48    45    79    86    10    47    67    86    64    38
[2,]   34   46   97   88   52   49   88   15   55     6    77    49    46    14    35    40     1    70    46    28    73    24    44    55
[3,]    8   37   38   30   20   68    4   78   41     2    62    38    95    82    96    56    61    34     1     9    57    80    58     7
[4,]   62   86   46    3   22   33   90   94   98     9    84    69    26    22    49    90    81    12    54    53    18    56    43    50
[5,]   10   60   34   96   79   62    6   15   94    39    85    17    55    59    48    11    63    98    33    49    41    82    12    61
      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
[1,]   13   69    4   21   25   74   11   85   50   21   93   98   91   26    4   28   61    8   55   68   30   26
[2,]   94   52   40   76   92   22   46   19   18   50   31   41   51   73   22   21   79   93   98   89   44   64
[3,]   56   31   92   84   25   20   65   36   67   96   35   59   76   66   83   37   33    2   95   20   84   43
[4,]   82   18   89   49   20   76   90   64   89   16    3   12   45   67   97   80   97   92   56   50   95   25
[5,]   94   80   84   18   68    6   98   47    6   55   73   70   56   46   50   90   89   88   50   99   41   36
      [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68]
[1,]   81   95   49    6   82   71   61   84   29   44   83   12   54   19   52   59   74   42   60   43   80   34
[2,]   93   66   14   10   36   36   69   79   90   51   81   16   28   26   97   33   77   68   28   46   16   96
[3,]    4   44   74   92   55   98    8   30   18   28   54   68   42   28   12   57   81   19   16   88   75   11
[4,]   52   21   47    4   67   62   63   25    3   55   87   84   28   73    8   38   80   53   70    9   87   33
[5,]   16   84   98   44   42   43   62   49   22   16   14   55   45   20   35   89    7   64   36    5   63    8
      [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90]
[1,]   74   46   40   27   79    1   98   44    8   55   55   27   69   79   97   39   54   75   88   87   97    4
[2,]   59   56    4   52   89   84   42   37   90   51   84   16   39   16   37   19   15   68    7   58   47   27
[3,]   57   99    4   51   26    6   16   71   52   96   89   79   91   57    2   33   42    1   40   76   85   99
[4,]   11   65   92   34   64   96   67   54   26   32   10   91   90   31    7    6   77   91   42    5   46   54
[5,]   76   95   40   51   98   77   92   35    3   56    1   17   70   45   87   37   70   84   69    7    5   97
      [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
[1,]   79   59    7   89   61   42   79   85   66    99
[2,]   82   72    1    7   22   40   59   16    7    72
[3,]   65   98   78   61   83   30   14   62   41    91
[4,]   56    1   24   62   47   63   39   63   95    45
[5,]   47   12   54   84   49   89   86   37   52    85
```

Figura 26. MeanMat1005 (m = 5 i n = 100)

```
$tij
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
 [1,]   52   95   42   75   44   57   89   53   84    62    91    14    95    89     4    95     2    97    68    20    33    51    98     8
 [2,]   82   21   79   95   46   23   40   95   87    37    24    24    65    62    19    67    66     6    65    59     2    67    82    90
 [3,]   63   55   59   35   21   59   78   25   30    38    78    79    58    44    38    76    70    72    85     8    10    84    42    67
 [4,]   79   21   93   32    8   45   37   78   26    98    17    25    21    28    68    24    62    89    60    64    38    90    87     1
 [5,]   80   13   64   77   17   78   82    4   72    93    68    25    67    80    43    93    21    33    14    30    59    83    85    85
 [6,]   53   19   99   62   88   93   34   72   42    65    39    79     9    26    72    29    36    48    57    95    93    79    88    77
 [7,]   71   90   59   82   22   88   35   49   78    69    76     2    14     3    22    26    44     1     4    16    55    43    87    35
 [8,]   95   53   54   22   84   54    2   80   84    66    25    16    79    90    51    29    29    90    83    83    19    95    87    12
 [9,]   92   62   11   83   87   66   98   42   23    45    52     6     3    64    55    97    83    42    81    92    68    46    56    88
[10,]   61   86   16   42   14   92   67   77   46    41    78     3    72    95    53    59    34    66    42    63    27    92     8    65
      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
 [1,]   85   86   73    4   40   98   12   59   44   46    2   41   28   83   28   21   80   71    4   60   34   55
 [2,]   30   63    5   93   53   85   81   73   34   74   13   78   35   20   16   48   12   11   80    9   24   76
 [3,]   20   24   75   23   33   60   20   75   83   26   92   29   39   14   74   66   86   10   27    8    7   97
 [4,]   99   34    9   22   74   14   14   84   75   37   32   29   32   89   12   47   19   97    7   12   43   89
 [5,]   70   35    2   76   46   72   69   46    3   57   71   77   33   49   59   82   59   70   76   10   65   19
 [6,]   94   39   74   46   17   30   62   77   43   98   48   14   45   25   98   30   90   92   35   13   75   55
 [7,]   76   98   78   81   48   25   81   27   84   59   98   14   32   95   30   13   68   19   57   65   13   63
 [8,]   34   23   44   30   82   83   42   56   89   38   96   10    3   53   97   11   65   47   76   22   17   14
 [9,]   50   13   23   13   49   18   50   94   71   64   31   21    2   63   58   36   64   52    8   94   51   36
[10,]   34    6   42   39    2    7   85   32   14   74   59   95   48   37   59    4   42   93   32   30   16   95
      [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68]
 [1,]   53   96   37   37   63   99   69   70   53   21   10   31   80   18    5   18   17   71   90   93   14   49
 [2,]   32   35   66   48   16   26   46   66   76   31   36    8   37   21    3   76   67    5   47   72   66   56
 [3,]   84   56   61    9   94   34   89   62   47   66   76   15   18   54   24   55   96   10   12   96   53   92
 [4,]   14   33   56   57   22    6   24   55   48   57   78    5   50   83   70   21   71   58   36   50   31   86
 [5,]   77   86   21   75   96    3   50   57   66   84   98   55   70   32   31   64   11    9   32   58   98   95
 [6,]   80   67    3   93   54   67   25   77   38   98   96   20   15   36   65   97   27   25   61   24   97   61
 [7,]   26   96   53   94   27   93   49   63   65   34   10   56   51   97   52   46   16   50   96   85   61   76
 [8,]   11   69   91   53    3   80   78   32   53   43   85   19   48   49   66   22   37   51   82   59   88   77
 [9,]   82   30   17   21   80   38   55   34   85   44   47   66   19   66   61   60   98   82   79   71   28   74
[10,]   58   12   95   21   74   38    4   31   62   39   97   57    9   54   13   47    6   70   19   97   41    1
      [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90]
 [1,]   52    7   78   57   41   75   98   93   33   75   68   33   60   82   24   99    4   97   24   50   55   91
 [2,]   95   49   47   26   81   56   76   66   36   53   26   52   29   36   68   21   71   61   71   69   28   86
 [3,]   77    6   91   14   41   30   85   17   23   60   76   39   85   10   65   15   55   41   28   93   88   27
 [4,]   29   30   93   49   83   89   44   38   62   45   22   85   39   98   56   68   84   77   67   53   46   24
 [5,]   25    4   45   60   87   31    1   96   22   95   73   77   30   88   14   22   93   48   10    7   14   91
 [6,]   75   92   73   21   29    3   96   51   26   44   56   31   64   38   44   46   66   31   48   27   82   51
 [7,]   30   90   42   88   37   43   88   91   14   63   65   74   71    8   39   95   82   17   38   69   17   24
 [8,]   19   32   52    9   96   23   64   22   37    3   52   44   11   21   85    6   40   68   30   35   58   31
 [9,]   27   33   13    9   12   51   16   49   83   48   13   78   96   77   68   88   77   76   73   92   72   87
[10,]   57   60   62   14   90   76   12   89   37   35   91   69   55   48   56   84   22   51   43   50   62   61
      [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
 [1,]   46   58   17   47   82    6   15   91   74    42
 [2,]   27   41   86   55   17   62   96   59   53    93
 [3,]   77   81   19   76   55   67   65    8   18    56
 [4,]   52   96    2   88   33   27   49   78   82    65
 [5,]    5   43   30   79   39   34   77   81   11    10
 [6,]   90   63   85   36   69   67   81   18   81    72
 [7,]   66   75   52   59    4   73   56   19   39    51
 [8,]   11   11    6   59   64   65   23   80   75    63
 [9,]   66   98   40   31   75   45   98   90    4    23
[10,]   10   87   99   40   91   64   62   53   33    16
```

Figura 27. MeanMat10010 (m = 10 i n = 100)

```
$tij
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
 [1,]   12   72   29   16   22   61   69   71   86    16    54    53    64    96    79    68    98    87    40    74    22    23    47    55
 [2,]   27   97   53   16   36   28   50   41   42    86     2    45    86    43    92    73    99    84    83    61     7     8    62     3
 [3,]   24   57   68   58   76   70   72   56   56    55    89    23    42    26    87    17    38    24    82    55    47     7    99    89
 [4,]   42   16   44   75   32   66   31    6    6    52     7    92    45    70    72    48    86    63    61    92    73    49    53    20
 [5,]    5   42   47   47   47   78   94   18   14    97    54    73    94    17    97     7    94    94    68     4    60    70    68    22
 [6,]   27   69   38   66   21   65    3    7   10    11    41    64     9    15    71    86     2    26    57    78    84    20    21     3
 [7,]   51    9   22   15   84   38   71   11   45    48    29    31    14    56    59    35     7    20    48    88    54    30    81     2
 [8,]   48   44   73   95   72   12   30   22   25    92     5    44    90    35    24    64    82     7    34    91    59     5    47    96
 [9,]   42   18   31   18   55   82   64   36   48    74    17    60    18    88    37    29    19    51    20     7    43    77    95    92
[10,]   31   49   78   23   37   34   43    7   69    97    15    38    80    78    89    95     1    18    11    11    92    99    81    79
      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46]
 [1,]    21    90    68    14    42    41    40    63    23    75     4    31    87    25    47    70     1    49    59    11    93    38
 [2,]    21    50    71    95    70    60     4    58    16    30    80    71    59    75    52    25    86    48    53    54    98    36
 [3,]    20    65     3    45    89    96    16    42    70    69    77    46    20    71    91     8    34    72    40    82    60    59
 [4,]    97    58    59    29    98    61    83     8    83    73     2    56     7    15    55     3    73    13    47    76    15    43
 [5,]    17    59    98    55    76     3    21    10    11    44    29    43    60    14    46     3    60    72    90     1    42    74
 [6,]    39    64    51    83    12    16     3     1    88    80    77     2    14    60    45    39    32    97    98    65    86     6
 [7,]     1    78     8     8    71    60    41    39     8    96     3    79    48    23    38    85    32    69    56    99    51    59
 [8,]     7    53    60    77    61    14    88    97    92    65    73    19    99    92    19    66    25    79    36    10    23    37
 [9,]    27    70    91    74    62    80    85     8    39    85    83    76    31    12    69    26    54    91    61    27    88    53
[10,]    53    47    48    32    58    33    65    86    60    47    89    40    72    54    29    57    93    65    39     8    87    57
      [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68]
 [1,]     5    43    82    33    26    89    86    10    16    72    96    47     9    56     8    69    87    84    62    52    82    66
 [2,]     9    78    80    88    52    30    62     8    45    77    90    52    39    95    86    15    98    59    22    43    85    93
 [3,]    59    43    72    10    63    67    12    41     5    25    67    95    46    37    41     2    25    99    48    24    18    10
 [4,]    44    13    16    18    16    18     3    11    98     9    28    44    14    39    76    92    59    95    11    55    56    16
 [5,]    54    41    15    38    24    91    49    23    63    88     1    30    85    22    94    39    70    85     7    65    43     4
 [6,]    14    99    93    54    54    92    57    39    13    70    73    16    43     9     2    76    38    66    21    32    78    41
 [7,]    47    79    67    83    58    49    50    19    64     5     5    52    15    26    67    11    77    74    88    83    96    73
 [8,]    76    83    58    99    29    35    49    89    79    82    92    45    42    21    51    30    70     2    20    82    13    92
 [9,]    47    39    14    78    90    90    50    64    94    79    10    18     7    26    51    23    43    95    57    30    86    80
[10,]     5    56    80     2    17    84    86    65    53     5    16    58    60    78    25    71    93    99     6    62    60    33
      [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90]
 [1,]    67    33     3    53    95    42    59    77    38    12     1    66    24    17    58    71    88    63    60    86     8     8
 [2,]     6    46    51    97    79    46    71    18    33    13     3    32    28    48     8    42    26    26    95    44    74    82
 [3,]    23    66    82    84    14    13    88     7    40    20    68    22    92    53    16    10    23    88    65    11    89    78
 [4,]    56    23    84    22    32    40    47    92    61    51    74    85     6    13    32     8    43    30    53    24     2    11
 [5,]    60    90    73    21    25    33    48    25    27    37    63    28    88    27    94    20    54    17    74    35    15    53
 [6,]    52    88    76    14    86    35    81    32    75    38    55    96    19    35    41    77    36    24    52    60    43    30
 [7,]    59    94    15    50    20    54    16    33     9    74    11    21    97    15    43     2     1    74    37    61     3    75
 [8,]    91     2    80    38    54    45    19    65    35    66    26    18    32    51    23    18    87    74    10    76    31    31
 [9,]    55    76    12    91    95    28    12    18    80    32    67     2     4    30    64    69    76    92    50    35    20    23
[10,]    57    79     7    69    82    26    41    52    38    21    84    74    76    16    75    75    36    79    71    82    70    22
      [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
 [1,]    44    99    80    76    94    75    63    57    76    34
 [2,]    28     4    42    81    56    38    50    12    43    51
 [3,]    94    32    64    42    74    87    63    94    87    72
 [4,]    33    27    35    40    40    96    69    23    32    13
 [5,]    64    38    79    45    67    93    16    59    47    95
 [6,]    86     6     8    31    80    82     2    59    47    21
 [7,]    41    86    51    65    42    50    48    88    52    46
 [8,]    93    61    26    39    83    11    57    81    40    97
 [9,]    52    25    82    18     2    29    70     7    36    96
[10,]    39     8    22    79    90    55    77     3    20    61
```

Figura 28. MeanMat10020 (m = 20 i n = 100)

Cal destacar que aquestes instàncies són originalment matrius de temps t_{ij} deterministes, de manera que cal adaptar aquestes dades al problema de flux de tasques amb temps estocàstics. Per a aconseguir aquest propòsit, les instàncies de Taillard anteriors seran considerades al present estudi com a matrius de mitjanes $MeanMat$ de les distribucions normals de temps de cada parella màquina/tasca. Per representar la variabilitat del problema, es crearà una matriu de variàncies que, juntament amb la matriu de mitjanes, servirà per generar finalment les matrius de temps que permetran calcular els valors de la funció objectiu.

Les matrius de variàncies $VarMat$ estan formades per tots els valors de les variàncies de cada distribució de probabilitat normal, per a cada parella màquina/tasca i han estat generades de manera aleatòria a partir d'un valor mínim i un valor màxim determinats per l'usuari, mitjançant la funció $VarMatrixGenerator$. Els valors mínim i màxim de les variàncies generades en el present estudi són $LowerBound = 0.1$ i $UpperBound = 0.2$, respectivament. Aquests valors han estat escollits seguint els valors de variàncies usats de manera habitual a la bibliografia consultada.

El procediment seguit doncs, per a generar una instància per al problema de flux de tasques amb temps estocàstics és el següent:

1. Càrrega de la matriu de mitjanes $MeanMat$, basada en les instàncies de Taillard.
2. Generació de la matriu de variàncies $VarMat$, a partir dels valors mínim i màxim $LowerBound$ i $UpperBound$.

A partir d'aquestes matrius, l'algoritme podrà generar les matrius de temps t_{ij}, agafant com a valors de μ_{ij} i σ_{ij}^2 els corresponents a les matrius de mitjanes i variàncies. En la Figura 29 es mostra el codi necessari per generar cada instància.

```
load("TaillardFS.RData")

#Màquines: m = 5 ; Tasques: n = 20
tai205 <- tai20.5[[1]]
MeanMat205 <- tai205$tij
VarMat205 <- VarMatrixGenerator(LowerBound, UpperBound, m = 5, n = 20)
```

Figura 29. Codi en R per a generar una instància del problema de flux de tasques amb temps estocàstics

4.2. Sintonització de paràmetres

Un punt de molta importància per al correcte desenvolupament de l'estudi és la sintonització de paràmetres. L'objectiu d'aquest estudi, tal i com s'ha repetit anteriorment, és el de comparar diferents heurístiques a l'hora de resoldre el problema de flux de tasques amb temps estocàstics. Per a poder fer aquesta comparació, és molt important que les heurístiques juguin en igualtat de condicions. Aquesta condició d'igualtat ve determinada pel nombre de vegades que un algoritme avalua la funció objectiu, que a la vegada, va lligat amb el nombre d'iteracions que es realitzen de l'algoritme de cerca local. Cada cop que es troba una solució candidata s'avalua la funció objectiu associada i, en funció del seu valor, l'algoritme guarda o no la solució i segueix el seu procés. Per tant, és de vital importància que els algoritmes avaluïn la funció objectiu el mateix nombre de vegades, per tal de que disposin del mateix nombre "d'oportunitats" de trobar una bona solució. En aquest joc, que consisteix en trobar bones solucions, qui troba una millor solució en un menor temps guanya.

4.2.1. Tabu Search

El primer pas per a sintonitzar els paràmetres de TS i SA és aconseguir una bona sintonització dels paràmetres de TS, a través dels quals després es podrà sintonitzar l'algoritme SA.

Per a sintonitzar el paràmetre $tabusize$, s'ha representat gràficament l'evolució del makespan a l'executar la funció $TSDet$ per al nombre d'iteracions $iter = 100$ i $tabusize = 2{,}7\ , 10\ i\ 50$. Per altra banda, es fan servir els mateixos gràfics per avaluar el nombre d'iteracions necessàries per a trobar una solució estable.

L'anàlisi dels paràmetres de TS es realitza en les Figures 30, 31 i 32, segons la dimensió de la instància $n = 20, 50\ i\ 100$, respectivament. Pel que fa al valor de $tabusize$ es pot observar que no té gran influència per a les instàncies amb $n = 50\ i\ n = 100$, però que per a $n = 20$ presenta una oscil·lació més accentuada a mesura que augmenta el seu valor. D'inici es descarten els valors de $tabusize = 10\ i\ 50$, a causa de l'oscil·lació que presenten. D'entre els valors restants que queden, s'escull el valor $tabusize = 7$, ja que presenta estabilitat i permet més flexibilitat

que un valor petit d'aquest paràmetre a l'hora de cercar solucions veïnes durant l'execució de l'algoritme.

Per altra banda, el nombre d'iteracions $iter$ per les quals la solució comença a estabilitzar-se és, observant les Figures 30, 31 i 32 $iter \approx 20$ per a $n = 20$, $iter \approx 30$ per a $n = 50$ i $iter \approx 50$ per a $n = 100$.

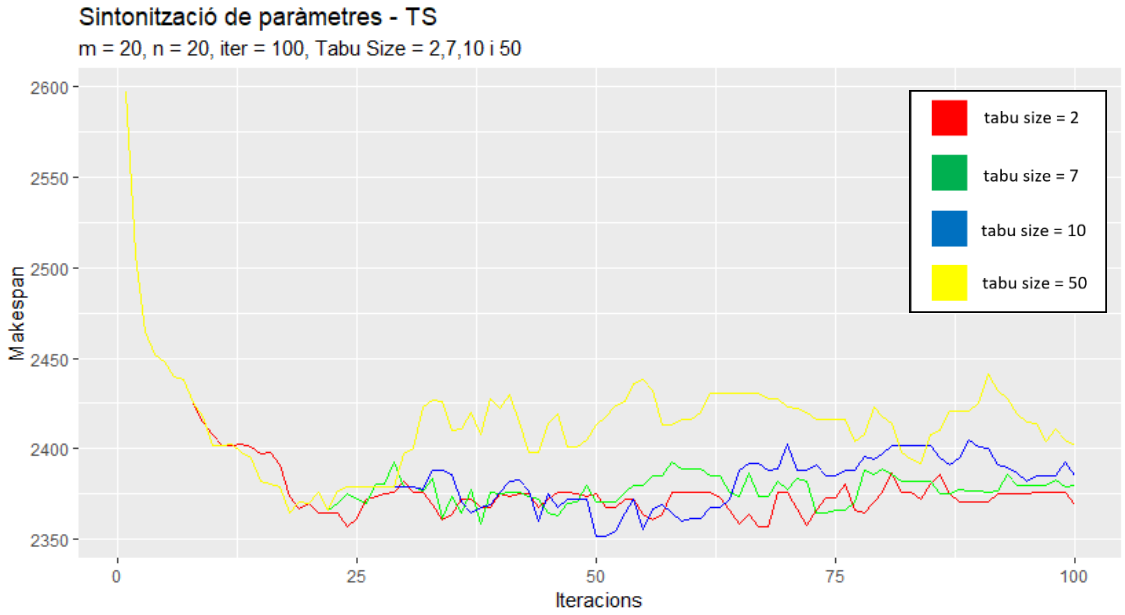

Figura 30. Sintonització dels paràmetres de TS per a m = 20 i n = 20

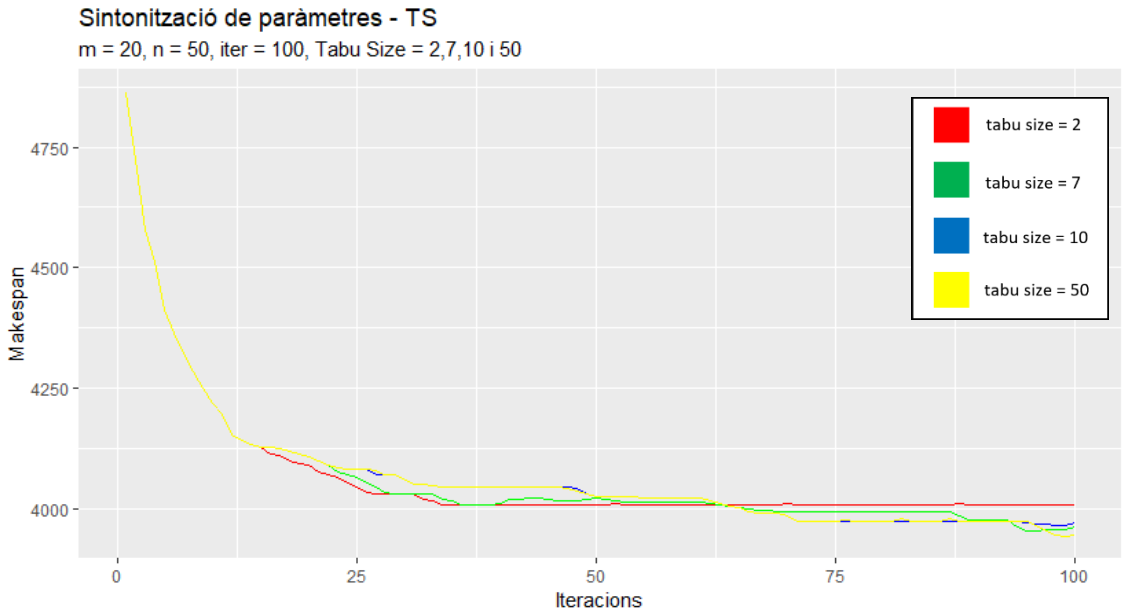

Figura 31. Sintonització dels paràmetres de TS per a m = 20 i n = 50

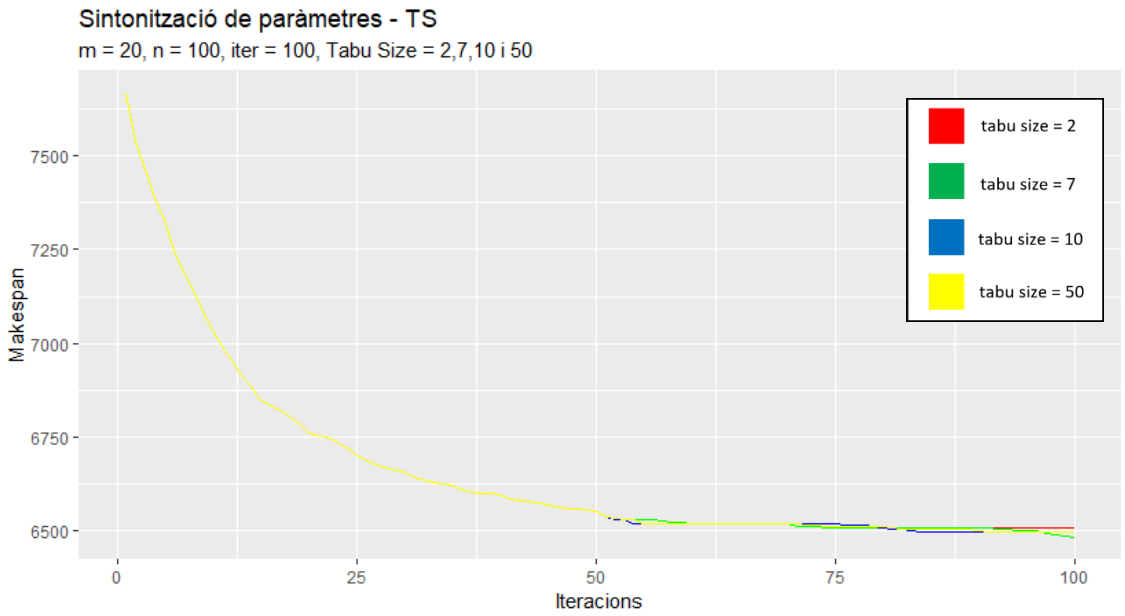

Figura 32. Sintonització dels paràmetres de TS per a m = 20 i n = 100

Així doncs, els paràmetres escollits per a executar l'algoritme TS són:

	tabusize	iter
n = 20	7	20
n = 50	7	30
n = 100	7	50

Taula 20. Taula resum dels paràmetres sintonitzats per a TS

4.2.2. Simulated Annealing

A continuació es procedeix a sintonitzar els paràmetres de SA, agafant com a referència els valors sintonitzats a TS. Això és així per assegurar la igualtat de condicions necessària per a comparar les diferents heurístiques. La condició d'igualtat, com s'ha comentat anteriorment, ve determinada pel nombre d'avaluacions de la funció objectiu, que es vol que siguin les mateixes tant per a TS com per a SA.

En el cas de SA, la solució candidata escollida és una solució veïna trobada mitjançant l'intercanvi aleatori de 2 nodes, de manera que a cada iteració de l'algoritme es realitza 1 única avaluació de la funció objectiu. Per altra banda, en el cas de TS, per a cada iteració de l'algoritme, es generen $(n-1)^2$ solucions veïnes amb els seus corresponents valors de la funció objectiu, d'on s'escull la solució veïna que presenta un millor valor de la funció objectiu. Per tant, es té la següent relació entre el nombre d'iteracions en SA (T_{max}) i TS ($iter$):

$$T_{max} = iter * (n-1)^2$$

Així doncs, donats el valors de $iter$ obtinguts a l'apartat 4.2.1, es pot calcular el nombre d'iteracions de SA necessaris per a que es produeixin el mateix nombre d'avaluacions de la funció objectiu que a TS:

$$T_{max}(n = 20) = 20 * (20 - 1)^2 = 7220$$

$$T_{max}(n = 50) = 30 * (50 - 1)^2 = 72030$$

$$T_{max}(n = 100) = 50 * (100 - 1)^2 = 490050$$

Un cop definits els valors de T_{max}, es pot procedir a determinar el valor del paràmetre μ. L'objectiu d'aquest paràmetre és aconseguir que l'algoritme es mantingui estable durant la seva execució i que existeixi equilibri entre la probabilitat d'acceptar una mala solució i la capacitat de l'algoritme per explorar bones solucions, tal i com s'explica a l'apartat 2.3.5.1 i la Figura 9. A continuació, a les Figures 33, 34 i 35, es determina quin és el valor òptim de μ per a les instàncies de dimensió $n = 20, 50\ i\ 100$, respectivament.

Com es pot observar a la Figura 33, per a les instàncies de dimensió $n = 20$, el valor de $\mu = 10$ té una oscil·lació molt pronunciada fins que convergeix, a més en un valor de makespan pitjor, de manera que queda descartat. El valor de $\mu = 100$ presenta un bon comportament però es considera que la oscil·lació es pot amortir encara més. És per això que s'escull el valor $\mu = 200$, ja que la oscil·lació del makespan es va reduint progressivament a mesura que augmenta el nombre d'iteracions. El valor de $\mu = 1000$ mostra un amortiment massa pronunciat, de manera que pot provocar que l'algoritme encalli en males solucions a l'inici de l'execució.

Per a les instàncies de $n = 50\ i\ 100$, es segueix el mateix raonament que per a les instàncies de dimensió $n = 20$, buscant un amortiment de la corba progressiu, ni molt pronunciat ni molt feble. Així doncs, analitzant les Figures 34 i 35 es decideix escollir els valors de $\mu = 2000$ per a $n = 50$ i $\mu = 13600$ per a $n = 100$.

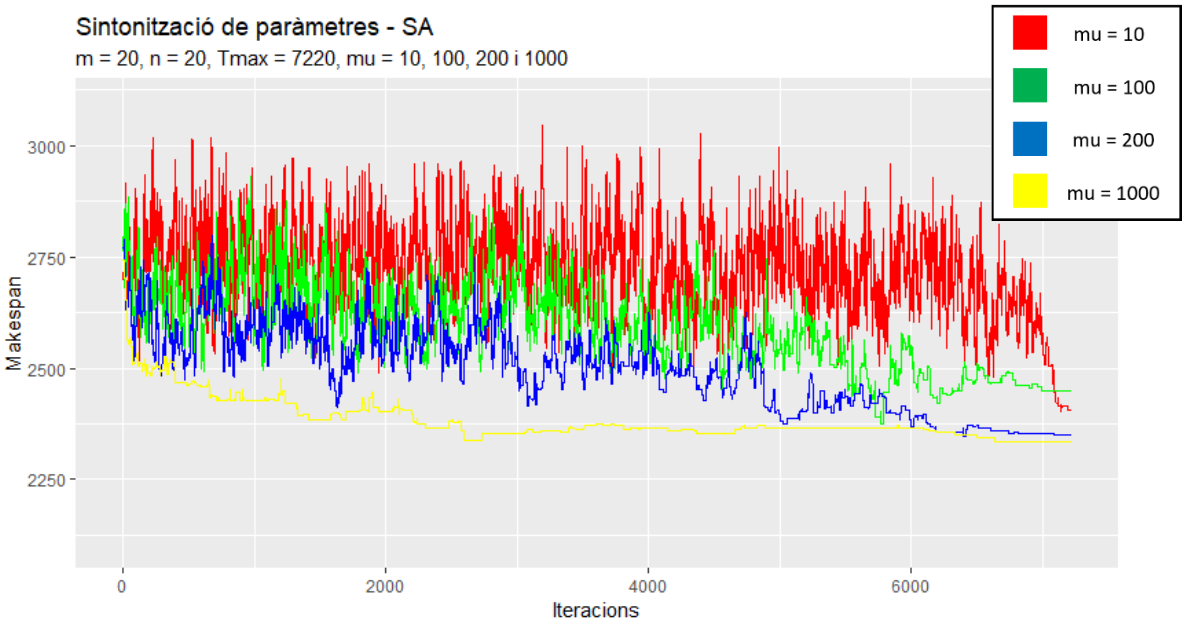

Figura 33. Sintonització dels paràmetres de SA per a m = 20 i n = 20

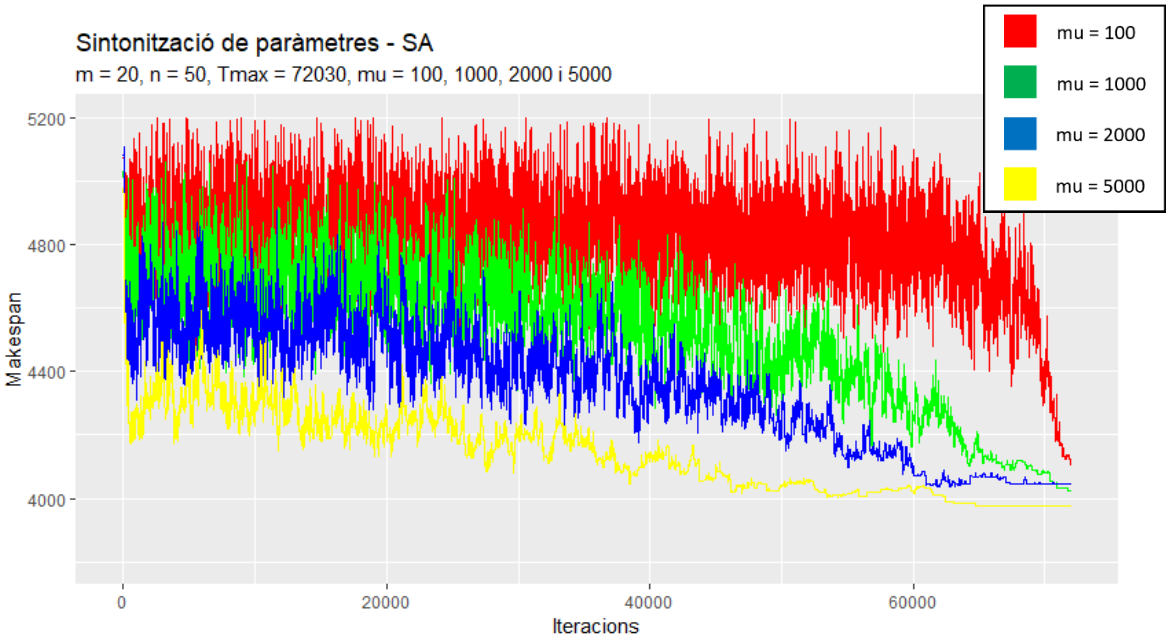

Figura 34. Sintonització dels paràmetres de SA per a m = 20 i n = 50

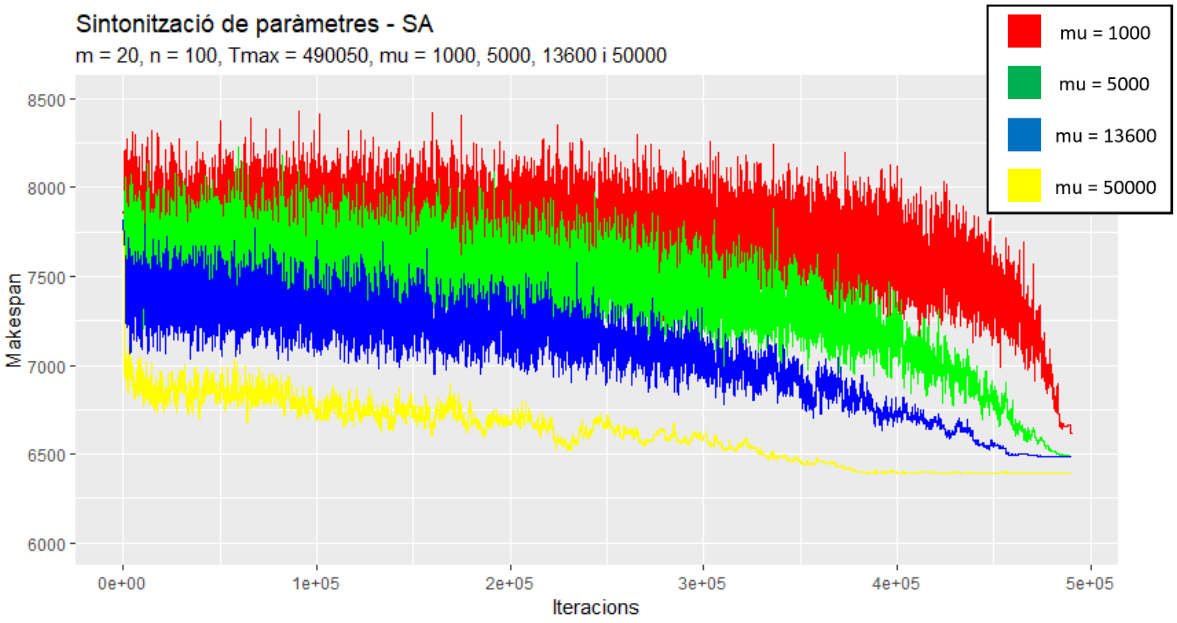

Figura 35. Sintonització dels paràmetres de SA per a m = 20 i n = 100

Així doncs, els paràmetres escollits per a executar l'algoritme SA són:

	μ	Tmax
n = 20	200	7220
n = 50	2000	72030
n = 100	13600	490050

Taula 21. Taula resum dels paràmetres sintonitzats per a TS

4.3. Resolució i tractament de resultats

Finalment, un cop sintonitzats els algoritmes de cerca local, es pot procedir a realitzar l'experiment computacional. Aquest es dividirà en 4 grans blocs, segons el tipus d'algoritme de cerca local i el tipus d'algoritme que calcula una bona solució inicial. Per tal de flexibilitzar el canvi d'algoritmes per a l'usuari, s'han afegit 2 variables (selectors) que permeten escollir els algoritmes dins la funció SIMILS. Aquestes variables són $LocalSearchSelector$ $(SA = 1; TS = 2)$, per a poder escollir l'algoritme de cerca local que es vulgui fer servir, i $InitialSolSelector$ $(NEH = 1; CDS = 2)$ per a escollir l'algoritme de càlcul de la solució inicial. Cada un dels 4 blocs serà executat pels valors de $m = 5{,}10\ i\ 20$ i $n = 20{,}50\ i\ 100$. Els 4 grans blocs en que es divideix el càlcul de resultats són:

- **SA+NEH:** En aquest bloc s'executarà l'algoritme SIMILS amb SA com a heurística de cerca local i NEH com a algoritme per a trobar una bona solució inicial. El codi usat per a aquest bloc es pot observar a la Figura 36.

```
resultsSANEH205 <- SIMILS(MeanMat = MeanMat205, VarMat = VarMat205, nREP, LONGREP, TREPBase, InitialSolSelector = 1,
                          LocalSearchSelector = 1, muValue = 200, IterMax = 7220, tabusize = 0,
                          m = 5, n = 20)
```

Figura 36. Codi per executar l'algoritme SIMILS per a SA i NEH

- **SA+CDS:** En aquest bloc s'executarà l'algoritme SIMILS amb SA com a heurística de cerca local i CDS com a algoritme per a trobar una bona solució inicial. El codi usat per a aquest bloc es pot observar a la Figura 37.

```
resultsSACDS205 <- SIMILS(MeanMat = MeanMat205, VarMat = VarMat205, nREP, LONGREP, TREPBase, InitialSolSelector = 2,
                          LocalSearchSelector = 1, muValue = 200, IterMax = 7220, tabusize = 0,
                          m = 5, n = 20)
```

Figura 37. Codi per executar l'algoritme SIMILS per a SA i CDS

- **TS+NEH:** En aquest bloc s'executarà l'algoritme SIMILS amb TS com a heurística de cerca local i NEH com a algoritme per a trobar una bona solució inicial. El codi usat per a aquest bloc es pot observar a la Figura 38.

```
resultsTSNEH2010 <- SIMILS(MeanMat = MeanMat2010, VarMat = VarMat2010, nREP, LONGREP, TREPBase, InitialSolSelector = 1,
                           LocalSearchSelector = 2, muValue = 0, IterMax = 20, tabusize = 7,
                           m = 10, n = 20)
```

Figura 38. Codi per executar l'algoritme SIMILS per a TS i NEH

- **TS+CDS:** En aquest bloc s'executarà l'algoritme SIMILS amb TS com a heurística de cerca local i CDS com a algoritme per a trobar una bona solució inicial. El codi usat per a aquest bloc es pot observar a la Figura 39.

```
resultsTSCDS205 <- SIMILS(MeanMat = MeanMat205, VarMat = VarMat205, nREP, LONGREP, TREPBase, InitialSolSelector = 2,
                          LocalSearchSelector = 2, muValue = 0, IterMax = 20, tabusize = 7,
                          m = 5, n = 20)
```

Figura 39. Codi per executar l'algoritme SIMILS per a TS i CDS

L'algoritme SIMILS un cop ha estat executat retorna la millor seqüència solució que ha trobat per al cas estocàstic. Aquesta solució és introduïda a la funció EMS i es realitza una simulació llarga del makespan esperat, de la qual se n'analitzen els principals valors estadístics relacionats, com són el màxim, el mínim, la mitjana i la variància dels valors de makespan simulats. Els resultats obtinguts es mostren a les Taules 22, 23, 24 i 25.

	SA + NEH			
	Màxim	**Mitjana**	**Mínim**	**Variància**
Taillard 20.5	1276	1267	1257	4,34
Taillard 20.10	1586	1574	1562	6,69
Taillard 20.20	2303	2290	2277	7,78
Taillard 50.5	2714	2699	2685	9,76
Taillard 50.10	3043	3028	3015	8,35
Taillard 50.20	3930	3914	3900	10,4
Taillard 100.5	5469	5443	5422	24
Taillard 100.10	5753	5732	5715	16,8
Taillard 100.20	6340	6317	6298	16,3

Taula 22. Resultats obtinguts per a SA amb NEH com a solució inicial

	TS + NEH			
	Màxim	**Mitjana**	**Mínim**	**Variància**
Taillard 20.5	1296	1285	1274	5,97
Taillard 20.10	1589	1578	1567	6,88
Taillard 20.20	2382	2370	2358	7,07
Taillard 50.5	2719	2704	2688	9,79
Taillard 50.10	3084	3067	3053	9,59
Taillard 50.20	3945	3929	3916	10,2
Taillard 100.5	5498	5475	5452	24,8
Taillard 100.10	5768	5747	5730	16,1
Taillard 100.20	6429	6409	6391	13,9

Taula 23. Resultats obtinguts per a TS amb NEH com a solució inicial

	SA + CDS			
	Màxim	**Mitjana**	**Mínim**	**Variància**
Taillard 20.5	1277	1267	1258	4,42
Taillard 20.10	1586	1576	1566	4,86
Taillard 20.20	2293	2279	2267	8,32
Taillard 50.5	2715	2698	2684	11,8
Taillard 50.10	3034	3019	3006	8,24
Taillard 50.20	3924	3905	3891	12
Taillard 100.5	5470	5444	5425	16,5
Taillard 100.10	5754	5732	5714	15,8
Taillard 100.20	6327	6303	6287	17,1

Taula 24. Resultats obtinguts per a SA amb CDS com a solució inicial

	TS + CDS			
	Màxim	**Mitjana**	**Mínim**	**Variància**
Taillard 20.5	1276	1266	1257	5,03
Taillard 20.10	1593	1582	1570	6,98
Taillard 20.20	2304	2290	2277	8,06
Taillard 50.5	2719	2703	2688	12
Taillard 50.10	3090	3075	3062	8,14
Taillard 50.20	3939	3922	3908	10,7
Taillard 100.5	5468	5443	5420	25,8
Taillard 100.10	5882	5862	5846	16
Taillard 100.20	6449	6425	6409	14,8

Taula 25. Resultats obtinguts per a TS amb CDS com a solució inicial

Les seqüències solució obtingudes per a cada cas es mostren a les Taules 26, 27, 28 i 29.

	SA + NEH
	Seqüència solució
Taillard 20.5	9 15 17 6 8 5 19 13 7 11 1 3 14 18 16 4 2 10 20 12
Taillard 20.10	18 5 9 12 17 3 4 2 8 20 13 15 10 6 19 11 14 7 1 16
Taillard 20.20	16 8 7 5 12 18 15 10 13 9 11 6 20 14 2 4 17 1 3 19
Taillard 50.5	31 41 28 8 24 50 17 39 27 11 2 25 34 30 18 15 26 42 21 20 19 29 9 13 45 23 4 22 3 6 1 14 46 38 43 16 10 5 32 47 33 44 49 7 35 40 48 12 37 36
Taillard 50.10	22 42 33 29 8 38 44 43 31 36 26 20 49 15 28 2 7 12 46 4 18 41 14 23 40 32 19 9 25 11 34 10 17 5 37 13 30 6 16 50 1 3 47 27 21 35 24 48 45 39
Taillard 50.20	35 31 39 47 24 36 15 10 20 27 1 7 33 5 46 40 34 43 44 9 45 2 6 32 42 29 11 12 16 23 14 38 49 37 41 17 22 26 13 21 8 4 48 18 50 30 19 25 28 3
Taillard 100.5	10 90 99 42 73 70 24 72 49 55 67 34 46 86 74 15 22 18 27 36 51 9 2 5 30 6 40 44 98 11 16 12 39 78 68 48 47 93 13 96 84 79 41 17 77 5 6 80 76 91 62 75 35 50 8 38 71 69 28 32 61 45 43 20 3 94 97 33 21 31 85 26 14 53 5 100 87 60 92 58 52 7 66 95 19 81 29 4 89 57 54 6 3 64 83 23 37 2 88 65 59 82 1
Taillard 100.10	70 21 15 77 17 62 58 61 85 41 44 80 81 95 31 74 96 67 90 63 42 6 6 9 49 79 97 2 87 38 89 39 76 82 93 29 35 64 34 1 10 24 52 20 7 14 60 22 27 73 55 91 19 88 78 43 3 66 36 5 98 48 84 68 57 92 83 13 3 2 46 28 23 71 11 37 65 56 40 94 86 33 53 9 8 4 47 50 75 18 72 51 30 99 16 26 25 100 54 45 59 12
Taillard 100.20	54 78 33 59 22 35 55 65 91 61 80 9 76 51 40 82 3 1 12 81 48 79 3 1 57 70 83 74 71 93 44 89 100 16 45 32 18 34 90 56 26 98 43 21 36 28 94 77 58 86 46 37 39 11 53 63 25 99 10 14 50 92 30 47 60 2 67 62 24 4 42 29 23 20 15 6 72 95 85 49 68 75 5 66 19 84 88 96 7 64 87 17 97 8 13 73 38 52 27 69 41

Taula 26. Seqüències solució obtingudes per a SA amb NEH com a solució inicial

	TS + NEH
	Seqüència solució
Taillard 20.5	17 13 9 8 15 4 14 11 16 19 5 3 1 6 7 2 18 12 10 20
Taillard 20.10	18 5 2 9 12 17 19 3 4 8 15 10 6 14 20 11 13 7 1 16
Taillard 20.20	4 16 7 3 17 1 6 18 12 15 10 11 13 20 14 9 2 5 19 8
Taillard 50.5	31 41 3 38 25 27 20 10 50 39 46 17 30 19 34 4 28 26 23 48 18 49 42 44 7 6 13 32 33 1 11 45 40 29 22 8 47 5 43 2 9 21 15 12 24 14 37 16 36 35
Taillard 50.10	42 33 18 30 37 34 44 15 38 3 2 20 22 49 23 4 25 16 12 36 17 32 35 43 19 6 14 21 27 47 13 31 29 41 1 50 26 48 11 39 46 8 9 28 7 10 40 5 45 24
Taillard 50.20	35 43 24 42 5 1 33 40 11 20 36 15 47 23 49 31 29 27 37 17 14 2 8 26 45 6 13 22 21 12 32 38 39 30 16 9 44 46 19 28 48 4 34 7 10 18 41 25 50 3

Taillard 100.5	31 84 40 55 93 60 77 71 90 20 81 70 41 46 59 9 61 25 65 24 72 100 7 82 10 83 80 33 29 86 58 35 34 79 73 76 92 16 87 64 23 28 26 32 19 39 21 85 14 99 95 98 53 96 74 13 8 54 4 3 62 15 97 69 38 6 89 36 68 51 49 42 17 78 45 18 94 27 52 48 50 2 11 75 47 63 57 37 67 44 22 91 12 30 43 5 1 56 66 88
Taillard 100.10	70 58 5 21 60 15 17 44 31 95 57 1 92 4 22 61 90 81 10 87 35 97 88 36 39 99 74 13 52 91 8 51 85 6 49 82 64 50 62 96 33 2 56 77 43 78 89 38 63 24 73 20 40 83 72 79 93 84 18 67 68 11 65 19 42 7 66 94 25 29 34 76 100 98 27 23 14 80 41 46 3 37 16 71 53 9 54 55 48 47 32 28 69 30 86 59 75 45 26 12
Taillard 100.20	54 78 22 33 82 83 89 59 31 10 51 88 79 36 35 55 1 21 25 37 92 74 77 100 80 5 29 44 75 85 65 73 11 4 94 12 47 38 40 14 90 57 60 6 48 7 30 32 76 87 98 45 61 24 43 3 91 46 86 18 39 53 20 81 84 28 56 58 9 27 62 72 93 16 97 42 70 17 34 71 13 95 23 99 2 52 68 64 67 19 63 69 15 8 50 49 96 66 26 41

Taula 27. Seqüències solució obtingudes per a TS amb NEH com a solució inicial

	SA + CDS
	Seqüència solució
Taillard 20.5	17 3 15 6 19 14 9 13 4 1 2 18 16 8 5 7 11 10 20 12
Taillard 20.10	18 5 2 12 9 10 15 4 14 8 19 17 3 13 6 20 11 7 1 16
Taillard 20.20	16 18 14 7 10 12 13 8 9 15 11 5 6 17 1 20 2 4 3 19
Taillard 50.5	31 10 40 28 23 26 42 41 4 34 45 46 17 44 38 29 30 48 9 18 15 25 32 27 7 5 13 6 49 3 43 35 2 22 39 11 24 14 1 47 20 50 16 8 33 21 12 19 37 36
Taillard 50.10	18 22 8 33 37 44 43 49 15 38 40 34 6 10 35 29 2 16 36 4 28 13 31 12 32 47 19 21 30 46 9 42 14 20 3 25 23 11 7 17 45 41 50 26 5 1 48 27 24 39
Taillard 50.20	35 43 31 34 39 47 49 37 8 44 1 27 45 17 5 11 20 40 29 28 14 10 6 42 7 24 15 33 18 26 46 19 36 32 16 2 23 38 13 12 22 4 21 48 9 41 30 25 50 3
Taillard 100.5	60 85 40 50 31 5 25 17 13 19 77 22 20 28 27 47 59 45 55 29 70 39 87 94 73 18 49 58 9 4 42 64 37 43 96 68 34 75 15 97 93 30 99 54 11 7 88 90 76 36 12 86 53 14 44 41 6 8 3 81 32 46 62 10 80 2 95 33 63 56 57 52 61 78 69 79 21 74 24 67 9 2 51 16 26 83 82 72 66 71 38 48 98 100 91 89 65 23 35 1 84
Taillard 100.10	70 21 77 61 58 15 83 62 64 17 39 80 97 49 41 6 98 91 31 1 25 60 96 90 44 14 2 0 74 81 55 93 57 9 42 2 89 94 73 23 63 65 84 69 22 67 50 68 5 35 92 10 76 18 6 6 24 71 8 88 32 28 85 3 7 27 79 29 43 82 51 38 46 13 33 37 87 4 30 75 78 95 16 19 48 52 34 100 11 59 56 36 54 47 86 53 99 26 45 40 72 12
Taillard 100.20	54 22 47 59 31 3 51 55 40 4 74 81 48 6 28 9 76 82 79 33 12 38 67 18 24 11 10 5 0 32 25 44 39 83 16 1 53 43 93 21 91 30 63 2 20 23 90 35 85 34 60 96 88 100 5 80 37 65 19 58 45 56 29 84 86 46 78 70 57 61 62 99 98 97 94 26 27 42 89 69 15 71 92 72 8 87 68 75 95 66 52 14 64 17 73 36 49 77 13 7 41

Taula 28. Seqüències solució obtingudes per a SA amb CDS com a solució inicial

	TS + CDS
	Seqüència solució
Taillard 20.5	15 6 2 13 14 11 17 8 7 1 19 4 3 5 18 16 10 20 12
Taillard 20.10	18 5 4 9 19 2 12 17 3 8 15 10 6 14 20 11 13 7 1 16
Taillard 20.20	16 18 14 7 13 5 10 8 15 20 11 6 12 9 2 1 17 19 3 4
Taillard 50.5	31 41 10 17 14 26 40 30 25 36 18 34 39 6 24 32 46 11 38 37 8 5 19 49 1 4 22 12 23 42 27 29 48 2 9 28 43 20 13 16 44 15 45 33 47 21 7 50 3 35
Taillard 50.10	18 44 25 20 30 42 33 37 34 24 49 43 31 15 2 1 22 4 27 38 8 14 3 29 9 40 16 6 26 48 12 32 36 7 28 23 46 13 47 35 10 11 50 17 41 5 21 19 45 39
Taillard 50.20	43 20 34 39 27 17 31 5 11 45 35 37 2 8 6 47 15 14 44 40 21 10 41 12 23 24 36 33 26 1 7 42 16 19 28 49 48 30 22 38 46 13 32 4 18 29 25 3 9 50
Taillard 100.5	74 40 99 90 11 50 10 93 42 83 19 31 58 25 60 28 34 29 38 46 72 48 17 77 45 12 7 100 24 84 71 13 64 96 66 56 76 75 70 20 15 49 33 4 61 59 85 43 78 79 3 62 6 8 65 41 53 95 8 23 9 5 39 6 21 44 26 81 52 88 30 63 69 86 27 73 82 91 97 67 47 14 51 57 2 54 32 98 18 80 22 92 87 94 37 36 16 55 89 35 1
Taillard 100.10	70 58 61 39 20 21 40 80 57 66 5 85 44 17 96 35 28 77 98 15 64 27 33 45 51 6 6 7 8 72 89 43 2 47 50 74 81 93 41 83 29 31 62 79 87 76 24 10 92 95 94 18 38 49 56 68 97 11 25 53 63 59 22 82 36 88 12 84 3 73 52 71 99 78 7 55 46 30 60 4 19 1 100 5 4 65 69 42 26 91 14 86 37 13 16 34 32 9 48 90 23 75
Taillard 100.20	22 54 47 78 89 77 3 40 82 1 25 65 60 31 48 11 61 79 32 74 90 2 83 51 4 41 35 3 6 85 59 30 53 76 92 80 86 8 46 97 58 45 50 33 93 81 62 5 21 55 39 100 68 37 7 3 18 9 44 38 15 6 28 69 88 91 72 7 94 57 12 99 75 56 20 70 13 67 24 10 43 14 9 8 71 87 16 23 42 52 66 19 95 29 26 63 17 34 96 84 49 64 27

Taula 29. Seqüències solució obtingudes per a TS amb CDS com a solució inicial

Per tal de poder comparar els resultats de manera visual per a les diferents heurístiques, es representaran gràficament les distribucions de probabilitat obtingudes per a les diferents dimensions de la instància d'entrada.

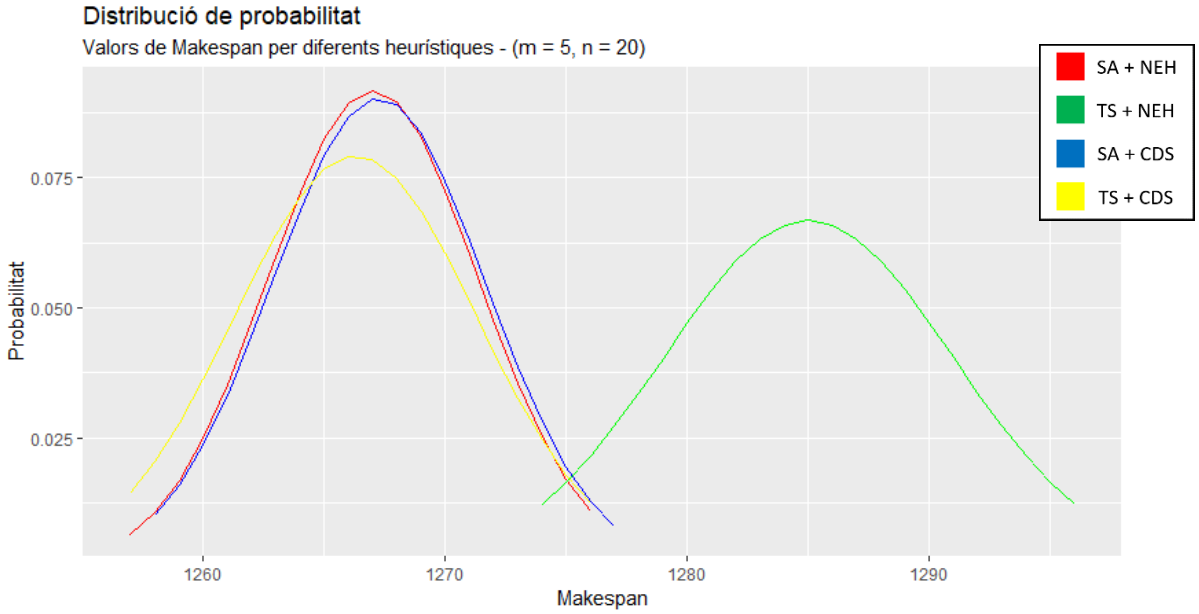

Figura 40. Comparació d'heurístiques per als valors m = 5 i n = 20

Com es pot observar a la Figura 40, els millors resultats s'obtenen al fer servir SA com a heurística de cerca local, sense apreciar-se diferències per a les solucions inicials NEH i CDS.

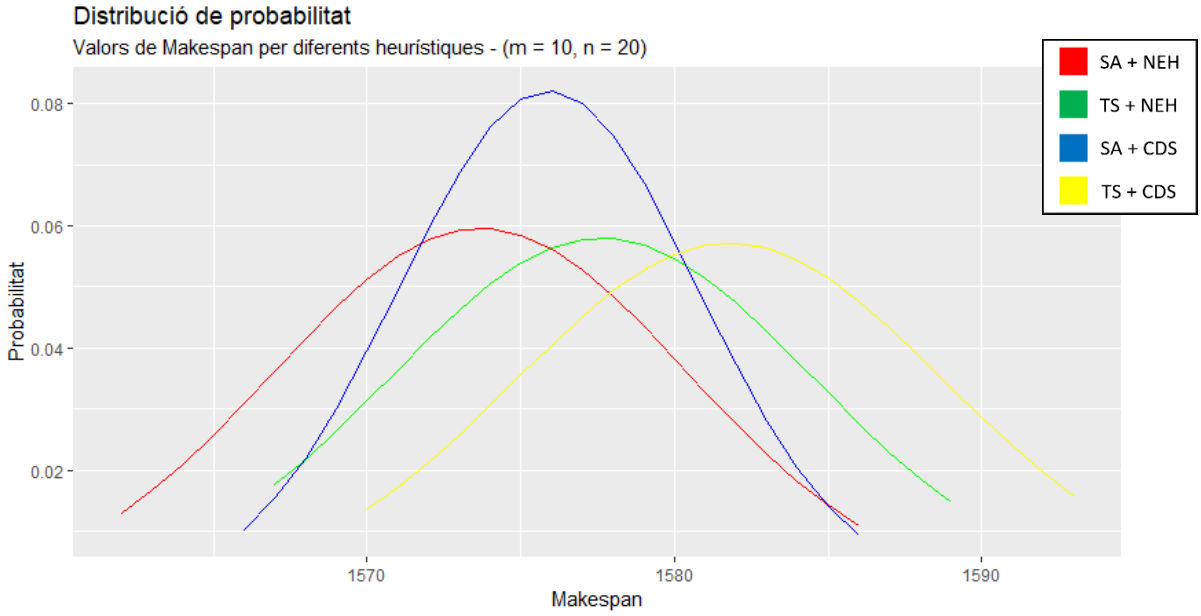

Figura 41. Comparació d'heurístiques per als valors m = 10 i n = 20

Novament, tal i com es pot observar a la Figura 41, els millors resultats s'obtenen al fer servir SA com a heurística de cerca local, però aquesta vegada els resultats són més ajustats, comparant-los amb els resultats obtinguts per a TS.

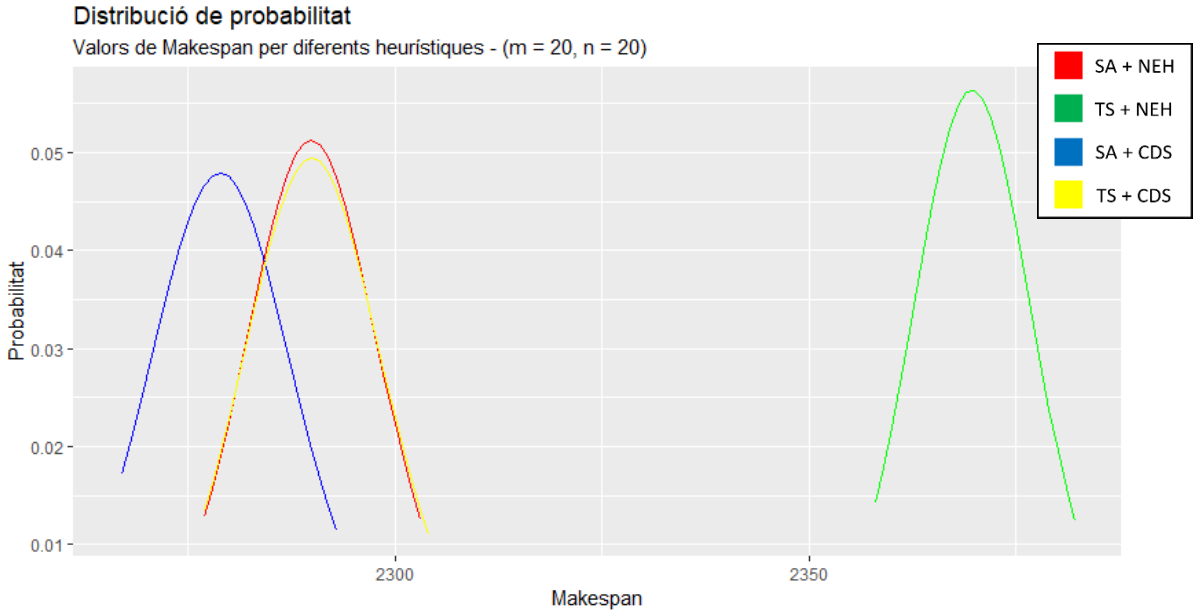

Figura 42. Comparació d'heurístiques per als valors m = 20 i n = 20

Per a la tercera simulació, com es pot observar a la Figura 42 es confirma la tendència dels resultats anteriors, sent SA l'algoritme que aporta els millors resultats.

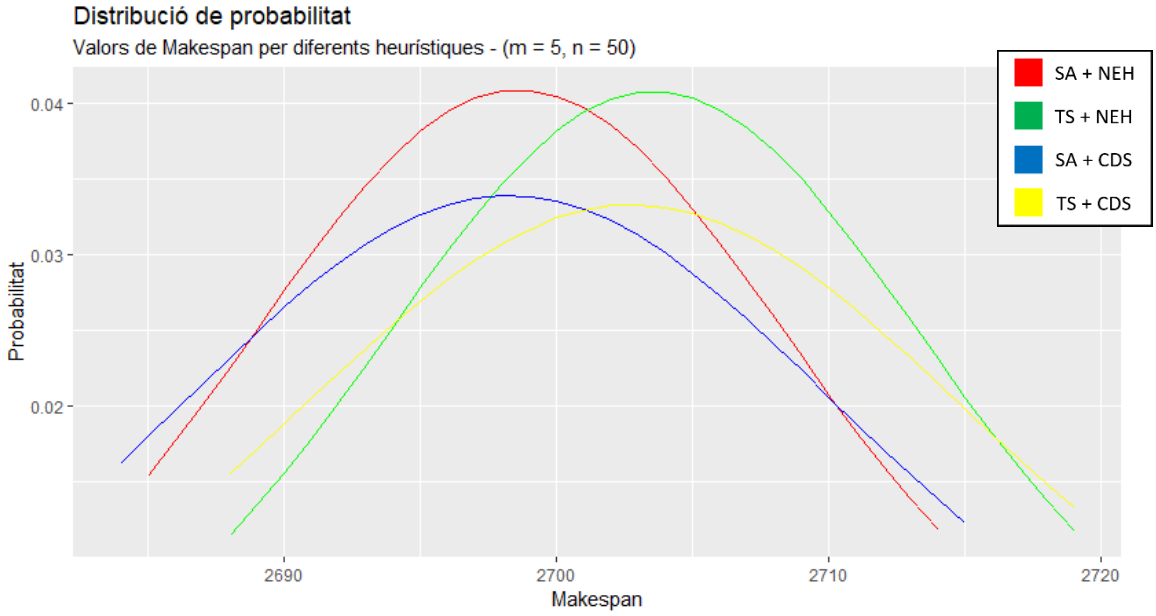

Figura 43. Comparació d'heurístiques per als valors m = 5 i n = 50

A la Figura 43 es pot observar com novament la heurística SA s'imposa respecte la heurística TS.

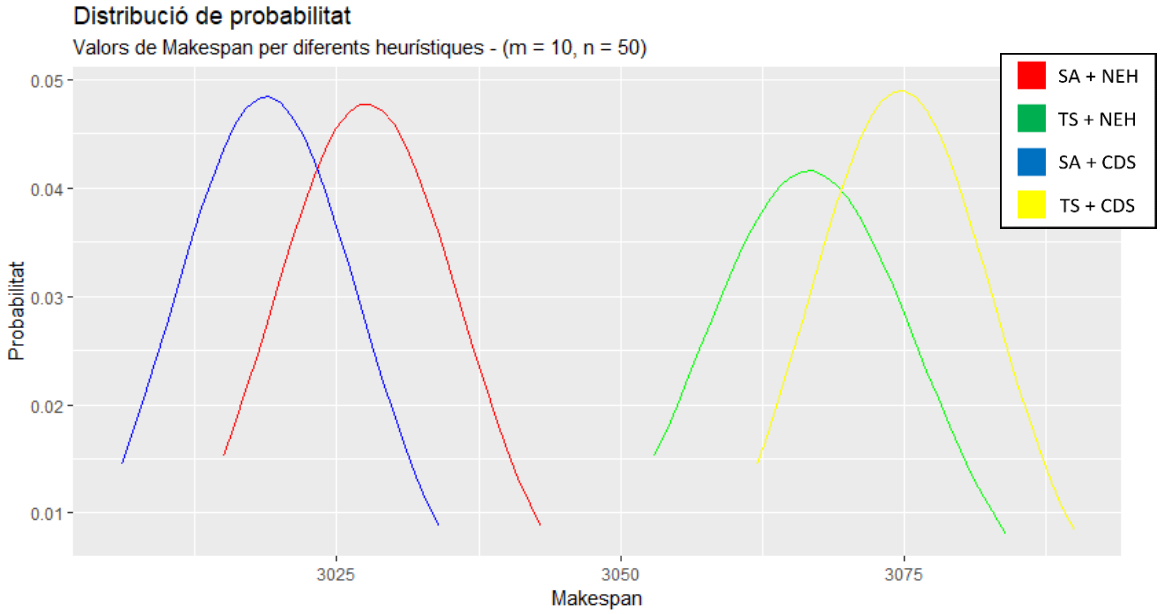

Figura 44. Comparació d'heurístiques per als valors m = 10 i n = 50

A la Figura 44, s'observa novament i de manera més clara, com l'algoritme SA produeix millors resultats que l'algoritme TS. Per altra banda, també cal destacar que es comença a veure una tendència de l'algoritme CDS a donar millors resultats que l'algoritme NEH pel que fa a la solució inicial aportada a l'inici de l'algoritme SIMILS, tal i com es pot observar en les tres últimes simulacions (Figura 42, Figura 43 i Figura 44).

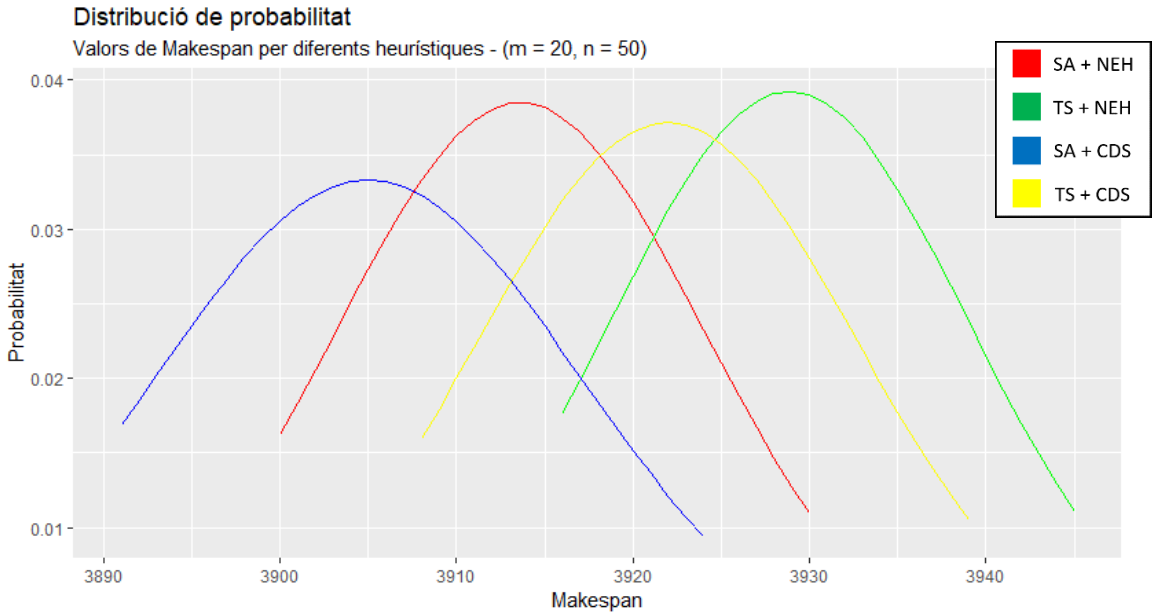

Figura 45. Comparació d'heurístiques per als valors m = 20 i n = 50

Novament, a la Figura 45, es torna a reafirmar el mateix comportament, on l'algoritme SA es torna a imposar i l'algoritme CDS reforça la seva tendència a aportar millors resultats de makespan.

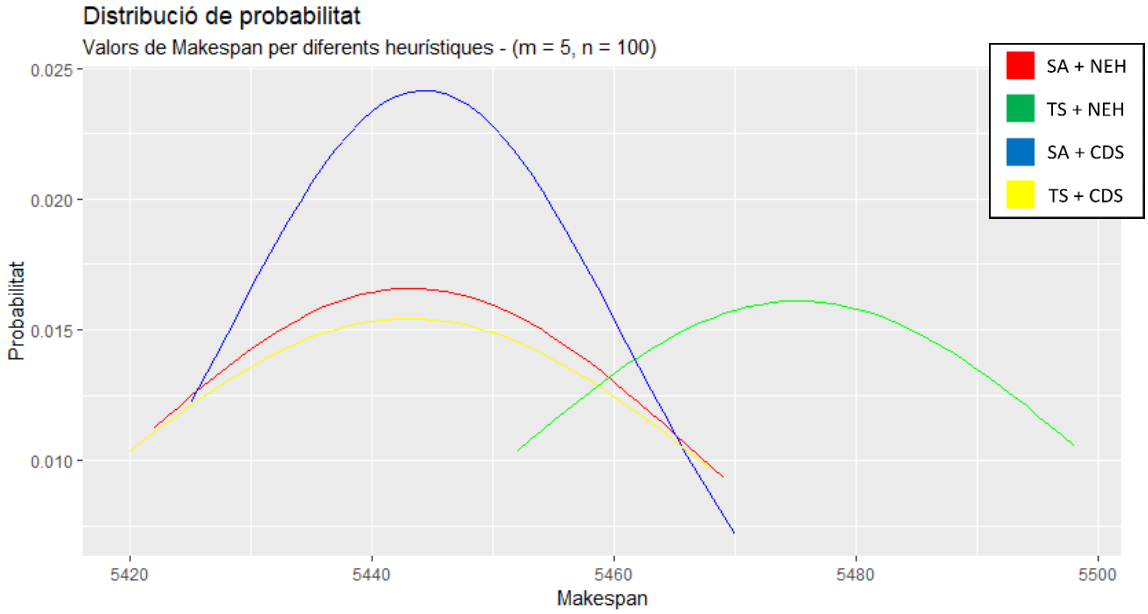

Figura 46. Comparació d'heurístiques per als valors m = 5 i n = 100

A la Figura 46 es tornen a obtenir els millors resultats per a l'algoritme SA, tot i que en aquesta simulació la combinació TS+CDS iguala els seus resultats.

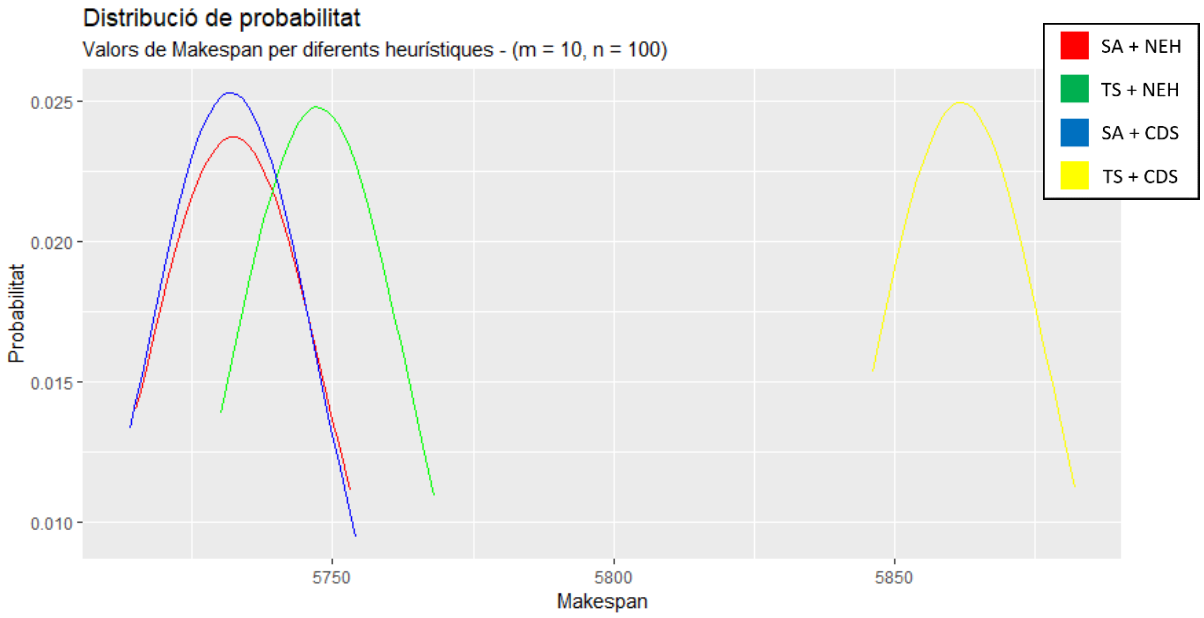

Figura 47. Comparació d'heurístiques per als valors m = 10 i n = 100

Altra vegada, es torna a produir el mateix resultat per a la simulació que es mostra a la Figura 47.

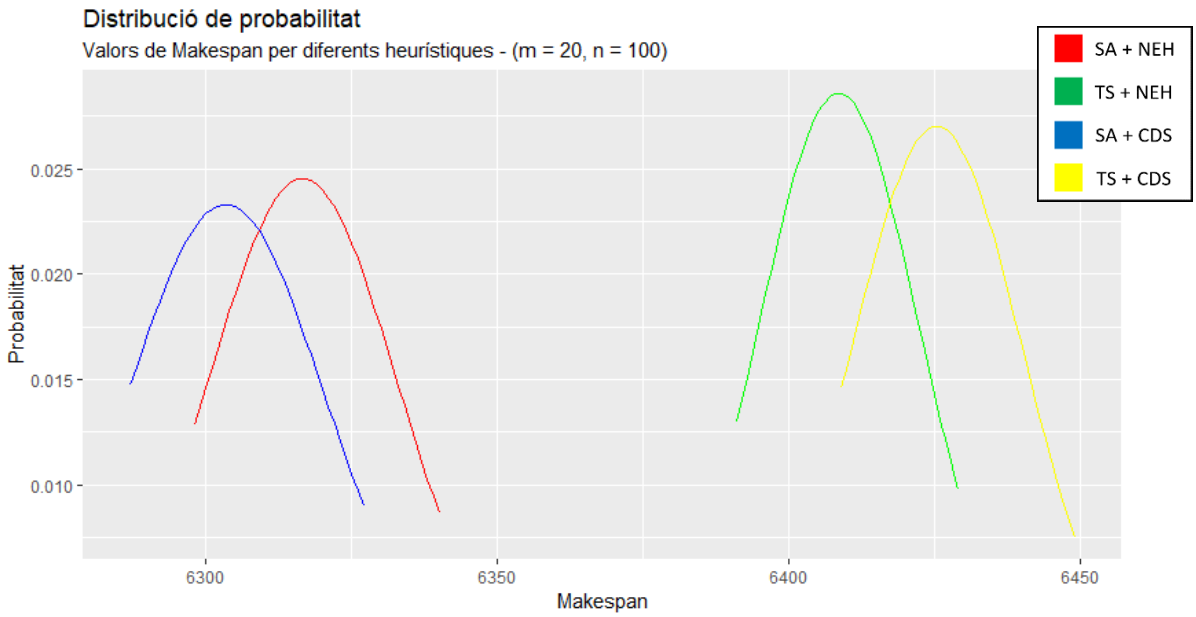

Figura 48. Comparació d'heurístiques per als valors m = 20 i n = 100

Finalment, la última simulació ofereix un resum de la globalitat de l'experiment computacional, a on s'ha imposat clarament l'algoritme de cerca local SA com el que aporta els millors resultats en el si d'un algoritme SIMILS. Per altra banda, tal i com s'observa a la Figura 48 i en general a la majoria de simulacions realitzades, no es detecta una gran influència de la solució inicial aportada a l'algoritme SIMILS, tot i que es pot observar com la heurística CDS s'imposa lleugerament a mesura que la dimensió de la instància augmenta.

4.3.1. Avaluació del temps d'execució

Finalment, per tal de completar l'anàlisi de les diferents heurístiques que s'ha dut a terme en aquest estudi, és necessari analitzar un factor important per valorar quin dels algoritmes és més eficient a l'hora de resoldre el problema del flux de tasques amb temps estocàstics. A part de la qualitat de la solució, que s'ha avaluat a l'apartar anterior, és necessari avaluar el temps de computació de cada algoritme.

Per avaluar el temps de computació de cada algoritme, s'ha fet servir el mateix paquet de R usat amb anterioritat a l'apartat 3.2. Es tracta del paquet microbenchmark, el qual permet executar 2 funcions diverses vegades i n'avalua la mitjana del temps que han trigat en executar-se.

Els primers algoritmes a ser avaluats són els algoritmes de cerca local SA i TS de manera independent, fora de l'algoritme SIMILS. Els resultats obtinguts són els que es mostren a la Figura 49.

```
Unit: milliseconds
                                                                                                  expr
    SADet(tai205$tij, sini = 1:20, Tmax = 7220, mu = 200, report.evol = TRUE)
 TSDet(tai205$tij, sini = 1:20, iter = 20, tabu.size = 7, report.evol = TRUE)

      min       lq     mean   median       uq      max neval
 97.80997 100.8514 103.43321 101.82603 103.08804 152.59806   100
 34.44330  35.2843  36.93092  35.90322  37.59575  81.53084   100
```

Figura 49. Resultats microbenchamrk SA vs TS

Com es pot observar a la Figura 49 l'algoritme SA triga substancialment més temps en executar-se que l'algoritme TS, per al mateix nombre d'avaluacions de la funció objectiu. Concretament, l'algoritme SA triga 2.8 vegades més temps en executar-se comparat amb l'algoritme TS.

D'altra banda, resulta interessant comprovar si aquesta relació de velocitat d'execució es segueix donant quan aquests algoritmes s'executen dins d'un algoritme global com és el SIMILS. A la Figura 50, es mostren els resultats obtinguts de comparar el SIMILS fent servir el SA com a algoritme de cerca local i el SIMILS amb l'algoritme TS com a algoritme de cerca local.

```
Unit: seconds

 SIMILS(MeanMat = MeanMat205, VarMat = VarMat205, nREP, LONGREP,
     SIMILS(MeanMat = MeanMat205, VarMat = VarMat205, nREP, LONGREP,
   TREPBase, InitialSolSelector = 1, LocalSearchSelector = 1,
       TREPBase, InitialSolSelector = 1, LocalSearchSelector = 2,
                                                                 expr
    muValue = 200, IterMax = 7220, tabusize = 0, m = 5, n = 20)
        muValue = 0, IterMax = 20, tabusize = 7, m = 5, n = 20)

       min        lq      mean    median        uq       max neval
 117.81541 117.85777 118.22470 118.25781 118.56073 118.63176     5
  52.72372  52.87984  53.34893  52.99329  53.89441  54.25338     5
```

Figura 50. Resultats microbenchmark SIMILS + SA vs SIMILS + TS

Com es pot observar a la Figura 50, la relació de velocitat d'execució del codi es conserva molt similar, en 2.2 vegades més gran l'algoritme SIMILS + SA front l'algoritme SIMILS + TS, per altra banda, un comportament esperat.

5. Conclusions

L'objectiu del present estudi és el de comparar diferents heurístiques a l'hora de resoldre el problema del flux de tasques amb temps estocàstics. Els algoritmes que s'han comparat han sigut els algoritmes de cerca local Simulated Annealing i Tabu Search, combinant-los amb les heurístiques NEH i CDS per a generar solucions inicials prou bones per optimitzar els resultats finals. La combinació d'aquestes heurístiques i metaheurístiques ha resultat en un algoritme que fusiona l'exploració de solucions deterministes amb la simulació de resultats estocàstics.

Per a analitzar i comparar aquests algoritmes, s'han tingut en compte 2 factors clau. El primer factor és la qualitat de la solució, radicada en el valor de la funció objectiu, que en el cas del problema del flux de tasques ve determinat per la minimització del valor de makespan . Per altra banda, el segon factor és la velocitat de computació, de manera que s'ha analitzat el temps d'execució de cada algoritme.

Per altra banda, la estructura bàsica dels algoritmes que s'han fet servir en aquest estudi està basada en els algoritmes existents per a resoldre el problema del flux de tasques amb temps deterministes. El pont entre el tractament d'instàncies deterministes i estocàstiques és la simulació dels valors de la funció objectiu. Per tant, la estratègia per millorar els algoritmes que treballen amb aquest tipus de problema es basa en realitzar simulacions de la funció objectiu en punts estratègics de l'algoritme i ajustar de manera correcte el nombre d'iteracions d'aquestes simulacions, per tal de reduir el temps de computació dels algoritmes, sense perjudicar la qualitat de les solucions.

Pel que fa a l'objecte de l'estudi, les conclusions que se n'extreuen són les següents:

1. S'observa de manera clara com la metaheurística Simulated Annealing, usada com a motor de cerca local dins de l'algoritme SIMILS, obté solucions de més qualitat que l'algoritme SIMILS fent servir la metaheurística Tabu Search com a motor de cerca local. La millora que es produeix al fer servir Simulated Annealing front Tabu Search oscil·la entre el 0,18-3,38 %, per al cas de fer servir l'algoritme NEH com a solució inicial i entre el 0,18-2,26%, per al cas de fer servir l'algoritme CDS com a solució inicial.
2. No s'observa una influència clara de la solució inicial sobre el valors obtinguts de makespan en el cas de les simulacions realitzades fent servir Tabu Search com a motor de cerca local.
3. A diferència del cas de Tabu Search, en el cas de fer servir Simulated Annealing com a motor de cerca local, s'observa la influència de la solució inicial sobre el resultat final, sent la heurística CDS la que proporciona solucions de més qualitat, a mesura que la dimensió de les instàncies augmenta.
4. Els temps d'execució de la metaheurística Simulated Annealing és de l'ordre de 2.8 vegades superior al temps d'execució de Tabu Search. Aquesta relació de velocitat es redueix sensiblement a 2.2 vegades superior quan aquestes metaheurístiques s'introdueixen dins l'algoritme global de SIMILS.

En vista dels resultats obtinguts, tant per a la qualitat de la solució com pel temps d'execució, es pot afirmar que l'algoritme SIMILS + SA és el millor a l'hora de resoldre el problema del flux de tasques amb temps estocàstics. Si bé obtenir una millor solució implica més temps, la diferència de temps d'execució és totalment assumible i està a l'abast de qualsevol usuari amb un equip informàtic comercial senzill.

Els resultats que s'obtenen en el present estudi queden restringits en l'abast d'aquest i no es poden extrapolar a instàncies amb dimensions més grans, per tant, una línia de treball futur és la d'avaluar el comportament dels algoritmes de SIMILS + SA i SIMILS + TS per a instàncies amb valors de $n > 100$.

Per altra banda, una altra possible via de continuïtat del present estudi és analitzar el comportament de les instàncies estocàstiques amb multitud de distribucions de probabilitat. En aquest estudi s'ha treballat amb temps distribuïts seguint una llei de probabilitat normal, però resultaria interessant conèixer el comportament de les instàncies enfront d'entrades que segueixen lleis de probabilitat exponencials, Weibull, Poisson, binomials, etc.

Un punt important a destacar, és la dificultat que ha presentat la realització del present estudi, a causa de la escassa bibliografia relacionada amb el problema tractat. Com s'ha comentat en diversos punts de l'estudi, el problema del flux de tasques amb temps estocàstics és un camp poc estudiat, de manera que la teoria relacionada està basada en la seva contrapart determinista i les vies d'anàlisi d'aquest problema es basen en l'experimentació. Així doncs, aquest estudi ha representat un repte tant a nivell conceptual com a nivell experimental, a l'hora de programar els algoritmes que s'han fet servir.

Finalment, es vol deixar constància que la feina que hi ha darrere d'aquest estudi i el codi desenvolupat es vol posar a disposició de qualsevol que vulgui consultar-lo, usar-lo, provar-lo i millorar-lo. És per això, que el codi que s'ha fet servir en el present estudi ha estat publicat a la plataforma de desenvolupament col·laboratiu GitHub. El repositori on es pot trobar el codi desenvolupat es pot trobar seguint el següent enllaç:

https://github.com/JordiAR/Flowshop-Problem-Heuristics/tree/master

6. Bibliografia

[1] RUIZ, Rubén; STÜTZLE, Thomas. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 2007, 177.3: 2033-2049.

[2] GOURGAND, Michel; GRANGEON, Nathalie; NORRE, Sylvie. A contribution to the stochastic flow shop scheduling problem. *European Journal of Operational Research*, 2003, 151.2: 415-433.

[3] PINEDO, Michael L. *Scheduling: theory, algorithms, and systems*. Springer, 2016.

[4] BAKER, Kenneth R.; ALTHEIMER, Dominik. Heuristic solution methods for the stochastic flow shop problem. *European Journal of Operational Research*, 2012, 216.1: 172-177.

[5] JUAN, Angel A., et al. A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, 2014, 46: 101-117.

[6] PAPADIMITRIOU, Christos H.; STEIGLITZ, Kenneth. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[7] DAWSON JR, John W., et al. The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Ingelligence, and Artificial Life plus The Secrets of Enigma, by Alan M. Turing (author) and B. Jack Copeland (editor). *The Review of Modern Logic*, 2007, 10.3-4: 179-181.

[8] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. & Stein, Clifford, (2010), Introduction to Algorithms, 3.ª edición, MIT Press and McGraw-Hill, (page 1049)

[9] DEAN, Walter. Computational complexity theory. 2015.

[10] DORIGO, Marco; GAMBARDELLA, Luca Maria. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1997, 1.1: 53-66.

[11] KORTE, Bernhard, et al. *Combinatorial optimization*. Heidelberg: Springer, 2012.

[12] The Travelling Salesman Problem. A: csd.uoc.org [en línia]. University of Crete, 2018. [Consulta: 2 abril 2019]. Disponible a: < https://www.csd.uoc.gr/~hy583/papers/ch11.pdf>

[13] GEETHA, S.; POONTHALIR, G.; VANATHI, P. T. A hybrid particle swarm optimization with genetic operators for vehicle routing problem. *Journal of advances in Information Technology*, 2010, 1.4: 181-188.

[14] RAD, Shahriar Farahmand; RUIZ, Rubén; BOROOJERDIAN, Naser. New high performing heuristics for minimizing makespan in permutation flowshops. *Omega*, 2009, 37.2: 331-345.

[15] SALLAN, Jose Maria. *Apunts "Introduction to metaheuristics"*. *Introduction to metaheuristics for optimization problems, 2018.*

[16] JUAN, Angel A., et al. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2015, 2: 62-72.

[17] JUAN, Angel A., et al. A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, 2014, 46: 101-117.

[18] Julia Micro-Benchmarcks. A: julialang.org [en línia]. Julia Lang, 2019. [Consulta: 8 abril 2019]. Disponible a: <https://julialang.org/benchmarks/>

[19] Benchmarks for basic scheduling problems. A: mistic.heig-vd [en línia]. E. Taillard, 1993. [Consulta: 10 abril 2019]. Disponible a: < http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html>

[20] R-Studio Support. A: support.rstudio.com [en línia]. R-Studio, 2018. [Consulta: 10 octubre 2018]. Disponible a: <https://support.rstudio.com/hc/en-us/>